2017

# Efficient cloud computing system operation strategies

Min Sang Yoon
*Iowa State University*

**Efficient cloud computing system operation strategies**

by

**Min Sang Yoon**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computing and Networking Systems

Program of Study Committee:
Ahmed E. Kamal, Major Professor
Guan Yong
Zhu Zhengyuan
Wang Lizhi
Akhilesh Tyagi



Iowa State University

Ames, Iowa

2017

# DEDICATION

I would like to dedicate this dissertation to my father Tae Eun Yoon, my mother Ok Rye Seo, my sister Seung Hee Yoon, and fiancee, Ha Neul Lee. I believe I could not be able to complete PhD degree without their support. Thank you for trust me and wait me for five years. I know it has been really long time and feel deep gratitude for their sacrifice. I also would like to express thanks to all my families and friends who support me.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGEMENTS

I would like to express my thanks to my adviser professor Dr. Kamal and those who gave me excellent instructions during PhD. First and foremost, I sincerely feel gratitude to Dr. Kamal for his devotional help for research, lecture, and care. I believe I could not complete PhD without his help. His endless passion in research and consistent effort in his work has inspired me and taught an attitude of authentic scholar. I also would like to thank my committee members for their help to this dissertation.

# ABSTRACT

Cloud computing systems have emerged as a new paradigm of computing systems by providing on demand based services which utilize large size computing resources. Service providers offer Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) to users depending on their demand and users pay only for the user resources. The Cloud system has become a successful business model and is expanding its scope through collaboration with various applications such as big data processing, Internet of Things (IoT), robotics, and 5G networks.

Cloud computing systems are composed of large numbers of computing, network, and storage devices across the geographically distributed area and multiple tenants employ the cloud systems simultaneously with heterogeneous resource requirements. Thus, efficient operation of cloud computing systems is extremely difficult for service providers. In order to maximize service providers' profit, the cloud systems should be able to serve large numbers of tenants while minimizing the OPerational EXpenditure (OPEX). For serving as many tenants as possible tenants using limited resources, the service providers should implement efficient resource allocation for users' requirements. At the same time, cloud infrastructure consumes a significant amount of energy. According to recent disclosures, Google data centers consumed nearly 300 million watts and Facebook's data centers consumed 60 million watts. Explosive traffic demand for data centers will keep increasing because of expansion of mobile and cloud traffic requirements. If service providers do not develop efficient ways for energy management in their infrastructures, this will cause significant power consumption in running their cloud infrastructures.

In this thesis, we consider optimal datasets allocation in distributed cloud computing systems. Our objective is to minimize processing time and cost. Processing time includes virtual machine processing time, communication time, and data transfer time. In distributed Cloud

systems, communication time and data transfer time are important component of processing time because data centers are distributed geographically. If we place data sets far from each other, this increases the communication and data transfer time. The cost objective includes virtual machine cost, communication cost, and data transfer cost. Cloud service providers charge for virtual machine usage according to usage time of virtual machine. Communication cost and transfer cost are charged based on transmission speed of data and data set size. The problem of allocating data sets to VMs in distributed heterogeneous clouds is formulated as a linear programming model with two objectives: the cost and processing time. After finding optimal solutions of each objective function, we use a heuristic approach to find the Pareto front of multi-objective linear programming problem. In the simulation experiment, we consider a heterogeneous cloud infrastructure with five different types of cloud service provider resource information, and we optimize data set placement by guaranteeing Pareto optimality of the solutions.

Also, this thesis proposes an adaptive data center activation model that consolidates adaptive activation of switches and hosts simultaneously integrated with a statistical request prediction algorithm. The learning algorithm predicts user requests in predetermined interval by using a cyclic window learning algorithm. Then the data center activates an optimal number of switches and hosts in order to minimize power consumption that is based on prediction. We designed an adaptive data center activation model by using a cognitive cycle composed of three steps: data collection, prediction, and activation. In the request prediction step, the prediction algorithm forecasts a Poisson distribution parameter $\lambda$ in every determined interval by using Maximum Likelihood Estimation (MLE) and Local Linear Regression (LLR) methods. Then, adaptive activation of the data center is implemented with the predicted parameter in every interval. The adaptive activation model is formulated as a Mixed Integer Linear Programming (MILP) model. Switches and hosts are modeled as M/M/1 and M/M/c queues. In order to minimize power consumption of data centers, the model minimizes the number of activated switches, hosts, and memory modules while guaranteeing Quality of Service (QoS). Since the problem is NP-hard, we use the Simulated Annealing algorithm to solve the model. We employ Google cluster trace data to simulate our prediction model. Then, the predicted data is em-

ployed to test adaptive activation model and observed energy saving rate in every interval. In the experiment, we could observe that the adaptive activation model saves 30 to 50% of energy compared to the full operation state of data center in practical utilization rates of data centers.

Network Function Virtualization (NFV) emerged as a game changer in network market for efficient operation of the network infrastructure. Since NFV transforms the dedicated physical devices designed for specific network function to software-based Virtual Machines (VMs), the network operators expect to reduce a significant Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). Softwarized VMs can be implemented on any commodity servers, so network operators can design flexible and scalable network architecture through efficient VM placement and migration algorithms. In this thesis, we study a joint problem of Virtualized Network Function (VNF) resource allocation and NFV-Service Chain (NFV-SC) placement problem in Software Defined Network (SDN) based hyper-scale distributed cloud computing infrastructure. The objective of the problem is minimizing the power consumption of the infrastructure while enforcing Service Level Agreement (SLA) of users. We employ an M/G/1/K queuing network approximation analysis for the NFV-SC model. The communication time between VNFs is considered in the NFV-SC placement because it influences the performance of NFV-SC in the highly distributed infrastructure environment. The joint problem is modeled by a Mixed Integer Non-linear Programming (MINP) model. However, the problem is intractable in large size infrastructures due to NP-hardness of the problem. We therefore propose a heuristic algorithm which splits the problem into two sub-problems: resource allocation and the NFV-SC embedding. In the numerical analysis, we could observe that the proposed algorithm outperforms the traditional bin packing algorithms in terms of power consumption and SLA assurance.

In this thesis, we propose efficient cloud infrastructure management strategies from a single data center point of view to hyper-scale distributed cloud computing infrastructure for profitable cloud system operation. The management schemes are proposed with various objectives such as Quality of Service (Qos), performance, latency, and power consumption. We use efficient mathematical modeling strategies such as Linear Programming (LP), Mixed Integer Linear Programming (MILP), Mixed Integer Non-linear Programming(MINP), convex

programming, queuing theory, and probabilistic modeling strategies and prove the efficiency of the proposed strategies through various simulations.

# CHAPTER1. INTRODUCTION

## 1.1 Background

### 1.1.1 Cloud Computing

Cloud computing [1], [2], [3], [4] is a new paradigm of computing systems sharing enormous computing and networking resources with multiple tenants. Multiple tenants can simultaneously use cloud computing resources by leasing Virtual Machines (VMs), containers, or physical servers depending on the purpose, and each of them can require heterogeneous amounts of CPU, memory, storage, and network resources. The resources should be provisioned to users immediately when needed, and released when no longer needing users. Cloud service vendors provides near infinite resources to users, and should be able to satisfy all types of user demands.

Since users pay for the used resources only, small enterprises or personal users do not need to spend budget for building their own infrastructures by utilizing cloud computing. Generally, traditional server utilization rates are kept below 20 to 30% by most server operators. Therefore, this causes waste of resources for most of the service operators. Cloud computing supports automatic resource scaling without human intervention. If the service operator hosts their servers or services on the cloud infrastructure, they can elastically control and provisions servers, storage or network resources to VMs depending on the various capacity demands of users. Therefore, the service operator can minimize the waste of computing resources and maximize their profit while meeting the required levels of quality of service.

Cloud services exist with various architectures: private clouds, public clouds, and hybrid clouds. Public clouds are typical cloud system perceived by most of the public. Clouds are run by third party companies such as Amazon Web Services (AWS) [20], Google Cloud [19], Microsoft Azure [23], and others. Public clouds are composed of multiple geographically dis-

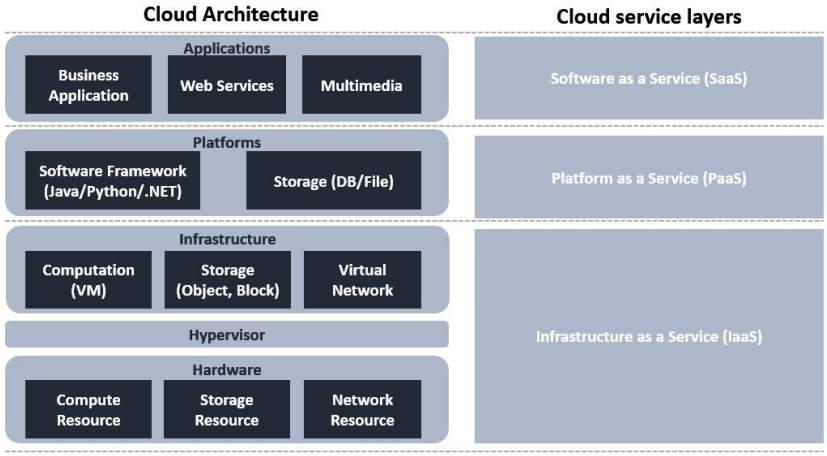


Figure 1.1: Cloud computing architecture and service layers

tributed data centers and networking devices connecting data centers. Physical resources are abstracted by computing, storage, and network resources, its resources are virtualized by using hypervisors such as KVM, Xen, or VMware vSphere. Since public clouds include multiple data centers, the configuration and provisioning of VMs or virtual networks are flexible in its architecture. Private clouds are more personalized cloud architectures built specifically to provide services internally to an organization. A private cloud architecture is supported by many solution providers such as IBM, VMware, or OpenStack. It is composed of CPU, storage, and network resources as well but they are not highly dispersed geographically. Since a private clouds is managed by a private manager, it provides a more secure environment compared to public clouds. Hybrid clouds are the combination of private and public clouds. By taking advantage of public and private clouds, hybrid cloud gives greater flexibility and more operational options for service providers. For example, the service provider can handle secure tasks in their private cloud and forward non-secure tasks to public clouds. Since the security issue is a controversial issue in cloud computing systems, a hybrid cloud is one of the appropriate solutions for customers who treat important data securely such as banks, investment companies, or insurance companies.

Fig 1.1 describes the basic architecture and service layers of cloud computing. Cloud systems basically support three types of service layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS delivers computing, storage, and

network resources over the network as user demands. Cloud service vendors provide resources as VMs, containers, or bare metal machines and network for inter-communications between reserved resources. Users can build their own service infrastructure on reserved resources. PaaS allows customers to develop, implement, and test software applications without the complex preparation of the operating systems and development environment. Users can still lease computing, networking, and storage resource depending on their needs but cloud service vendors support development platforms as well (e.g., Java runtime, NET runtime, etc). SaaS enables users to deploy application software in the cloud. Instead of installing and managing software in users infrastructure, users easily access this software via the Internet and the application software. SaaS applications are run on the cloud vendors' infrastructure and they manage access to the applications including security, availability, and performance.

In order to improve the performance of cloud computing infrastructures, cloud service providers actively adopt new technologies such as Software Defined Networks (SDN) and Network Function Virtualization (NFV) paradigms [8].

### 1.1.2 Software Defined Networks (SDN)

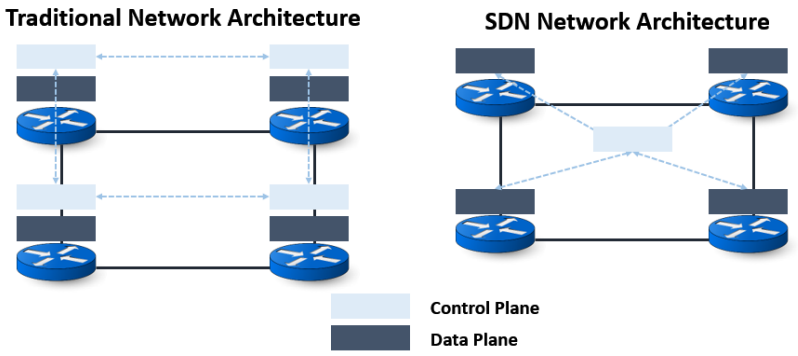


Figure 1.2: Traditional and SDN network architecture

SDN [5] is proposed to enable cloud, network engineers, and administrators to manage networking environments more easily and more optimally through centralized control. Fig. 1.2 describes the difference between traditional network and SDN network architectures. Unlike traditional network switches, SDN decouples the control plane and data plane. Traditional

networks have a static architecture in which the control and data planes reside in each network switch. Therefore, network switches make distributed decisions and this prevents dynamic, scalable, and flexible network management. However, the centralized SDN controller has a global view of the whole network, so the controller serves as a federation of multiple controllers, network resource slicing, network resource management, or routing path calculation.

### 1.1.3 Network Function Virtualization (NFV)

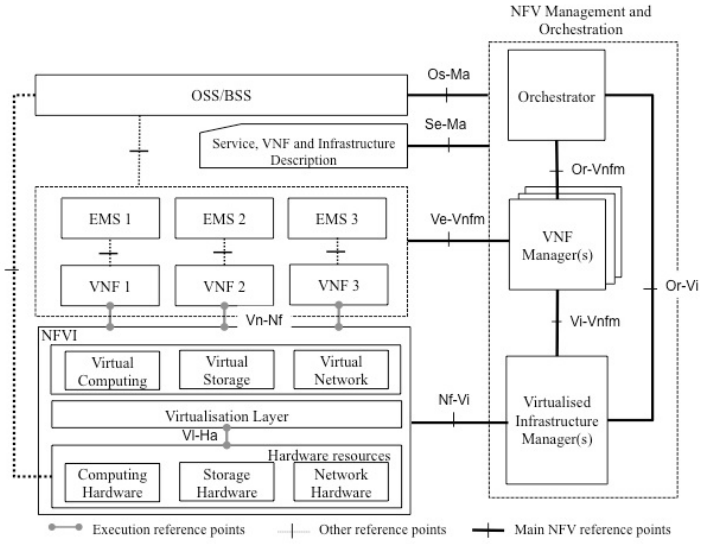


Figure 1.3: NFV architecture (taken from [82])

NFV [6],[7] is proposed by the European Telecom Standards Institute (ETSI) with the aim of efficient network architecture and network system operation. NFV transforms traditional network functions, such as Firewalls, QoS functions, and WAN optimizaton, implemented in dedicated physical machines to softwarized VMs which can be run on any commodity server built for general purpose use. Since NFV replaces physical network devices by VMs, the service providers expect to reduce CAPEX and OPEX significantly. Also, this enables a flexible network configuration because the service providers can migrate Virtualized Network Functions (VNFs) when they can achieve higher operational advantages. In order to overcome performance issues, they can easily control the capacity or how many of VNFs are operating depending on traffic demand.

Fig. 1.3 represents NFV architecture proposed by ETSI NFV [82], which has a similarity with cloud computing infrastructure. Network or cloud administrator have distributed computing, storage, and network infrastructure and its resource can be utilized for NFV services through the resource virtualization. Virtual Network Function (VNF) is a specific software network function such as Firewall, Deep Packet Inspection (DPI), or QoS functions. NFV service providers form a service chain for providing services to users. The service chain is a connection of multiple VNFs having a sequence of processing. Its architecture has a flexibility and reusability. In other words, a certain VNF can be used for multiple service chains and service chain can compose its architecture by placing VNFs to distributed infrastructure. NFV Management and Orchestration (MANO) components enable overall management and configuration of NFV services. NFV MANO is composed of three components: Virtualized Infrastructure Manager (VIM), VNF Manager (VNFM), and NFV Orchestrator (NFVO). VIM controls and manages the NFVI computing, storage, and network resources within one infrastructure sub-domain. In distributed environments, each domain includes the VIM and they independently manage NFVIs. NFV Orchestrator and VNFM collect information and monitor overall state of NFVI and control the performance of VNFs. VNFM controls life cycle management of VNF instances. Depending on user requests or network state, they can coordinate VNF resources. NFVO works for global resource management, validation, and authorization of NFVI resource requests.

## 1.2 Advantages of Cloud Systems

Cloud systems bring a lot of advantages in utilizing computing resources. Fundamentally, it changes the way that IT services are delivered to organizations or users. Instead, organizations own and manage their infrastructure in traditional IT environment, deploying cloud infrastructure can help customers to meet their IT requirements with a flexible, on-demand, and scalable. The main advantages of cloud systems can be summarized as below.

· **Cost**: organizations can reduce the Operational Expenditure (OPEX) and Capital Expenditure (CAPEX) by using cloud systems. Since organizations do not need to spend budget for building their own IT infrastructure, it is obvious that using cloud systems will save a significant CAPEX. Also, organizations can reduce OPEX because cloud systems automatically

manage the system and maintenance of the system are operated by third party companies.

· **Flexibility**: cloud systems include near infinite CPU, memory, storage, and network resources. Thus, customers easily configure and set their virtual infrastructure as they need. Also, customers can scale up or down requested resources depending on their needs.

· **Automation**: one of the fundamental objects of cloud systems is system automation. Cloud systems should be able to provision VMs, instances, physical devices, or virtual networks without human intervention. This advantage causes a significant efficiency to cloud service vendors.

· **Sustainability**: through leveraging economies of scale and the capacity to manage resources more efficiently, cloud computing consumes far less energy and other resources than a traditional IT infrastructure. This benefit can be achieved through efficient VM consolidation of load balancing strategies across the distributed cloud infrastructure.

Cloud service operators are making a significant profit by utilizing cloud computing service platforms. According to the recent report, Amazon's revenue generated from cloud-based Amazon Web Service (AWS) already has exceeded their profit from the traditional online market business. Many major IT companies such as Google, Microsoft, and IBM actively provide cloud computing based services and solutions and make a huge profit from them. Cloud service platforms also emerge as promising solutions for Internet of Everything (IoT), 5G network, big data processing, and self-driving vehicles. So cloud computing's importance and impact on future network and computing business are attracting many companies and researchers interest.

## 1.3 Research Challenges

Many challenges are faced by cloud service providers for efficient infrastructure operation. Since cloud service providers operate a myriad of computing, network, and storage devices in geographically distributed environments, efficient operation of the infrastructure is highly enforced by them for service quality, user experience, and operators' revenue maximization. First of all, power consumption of the cloud computing infrastructure is an extremely important issue. Since cloud service vendors are running multiple data centers and network devices

for connecting geographically distributed data centers, this consumes a significant amount of energy for operation. In 2014, U.S. data centers consumed about 70 billion kilowatt-hours of electricity which corresponds to 2% of the country's total energy consumption. Since traffic requests to cloud computing and data centers are explosively increasing, it is expected that the power consumption of the cloud computing will soar in the near future. According to [81], communication technology could use as much as 51% of global electricity in 2030 if not enough improvement is achieved in electricity efficiency of communication technology. With the goal of standardizing energy-aware network management, the European Telecommunications Standards Institute (ETSI) published ETSI Standard (ES) 203 237, the Green Abstraction Layer (GAL), which is a framework to manage energy efficient networks of the future [61], [62], [63]. GAL is an architectural interface that provides information exchange between power managed data plane and the control plane. GAL supports discovery, provisioning, and monitoring of functionalities in the networks, and regulates QoS of the network and the overall network power consumption through adaptation of Network-wide Control Policies (NCP) and the Local Control Policy (LCP).

Also, the cloud infrastructure has a highly distributed infrastructure worldwide. Utilizing such a distributed infrastructure brings benefits in terms of distributed processing, reliability, resilience of the system, and so on. However, several challenges arise from this distributed computing environment. VMs and virtual link allocation problems are representative problems in distributed cloud systems. Some users request large numbers of VMs and virtual networks from the cloud computing infrastructure. If multiple users access virtual networks from distributed regions, it is not easy to decide how to embed the virtual network onto the substrate network because the placement of virtual infrastructure affects service qualities of users and cloud infrastructure operation. In the cloud environment, VMs or virtual network resources are specified by users' requests and that include CPU, memory, storage and network bandwidth. However, the specific VMs and virtual network placement problem still remains to be solved for cloud service providers. Cloud service providers need to decide how to place the specific virtual machines or virtual networks onto their infrastructure so that they can minimize OPEX and maximize the performance of the infrastructure. This problem is a well-known NP-hard

problem, so optimal operation of distributed infrastructure is an extremely hard problem for service providers.

Resource allocation is also a challenging problem in cloud infrastructures. Users reserve VMs, containers, or physical machines based on their needs. Service providers need to assign heterogeneous VMs or containers to physical servers while guaranteeing the service quality of users, capacity of servers, or minimizing OPEX of the service provider. If VMs are highly consolidated in a single server or data center, it causes high utilization of physical servers. So the performance of physical servers can be degraded remarkably. In contrast, if the VMs are highly spread across the wide area of the infrastructure, long communication times and high cost are incurred between VMs. Since reserved service resources by tenants generate virtual networks, efficient virtual network resource embedding is also required to be considered as part of the resource allocation problem.

## 1.4 Dissertation Contribution

In this dissertation, we propose an efficient cloud system operation strategy in terms of various objectives: deployment cost, service time, power consumption, communication time, and Service Level Agreement (SLA) with the deployment of trending SDN and NFV technologies.

In chapter 3, we propose a Big Data processing scheme in distributed heterogeneous cloud environment. The cloud computing platform has emerged as a promising big data processing platform because of its near infinite resources. For example, Google Map-reduce programming model enables large size data sets to be processed in distributed clusters by dividing them into smaller pieces of data sets and this strategy is practically implemented in many applications such as Hadoop Distributed File System (HDFS). However, if we distribute sub-data sets in extremely distributed environments without insightful consideration, this will cause high levels of sub-data set transmission time and communication time. Big data has very large size data sets. Even if we divide them to sub-data sets, its size will be on the Terabyte or Exabyte scales. Transmitting this scale of data sets consumes significant transmission times and cost even if we utilize high bandwidth network environments. Concurrently, if we assign the sub-data set to consolidated clusters or servers for reducing transmission and communication time,

it will cause the processing time to increase and lose the chance to reduce the processing time and cost by utilizing the benefit of distributed cloud environment. Thus, an efficient sub-data set distribution strategy is considered for minimizing big data processing cost and time. We model this problem by using multi-objective Linear Programming (LP) that minimizes data set processing time and processing cost, and find the Pareto front using efficient searching algorithms.

We also consider the data center power consumption minimization strategy with user request prediction in Chapter 4. As described in the previous Section, the power consumption is an extremely important issue for cloud computing service providers. According to [39], data centers spent 91 billion kWh of electricity and 50% of the energy is wasted in the data centers due to the lack of awareness of the traffic in 2013. This data shows how the accurate prediction of the request plays an important role in data center power saving. We propose an advanced interactive adaptive resource activation model that jointly controls network switches and servers simultaneously while satisfying QoS of the data center based on user request prediction. For the prediction step, we collect the history of data requests of data centers and estimate the probability distribution of user requests. Based on the user request prediction for the following time interval, the data center controller decides the number of activated switches and servers. The server activation can be determined while satisfying requirements of the user requests. However, activating network switches requires precise connectivity control for the intra-data center communication. We model an optimal SDN controlled data center using Mixed Integer Linear Programming (MILP). This problem minimizes the operation power of the data center by deciding active switches, servers and operating memory modules in each switch and host. In a network switch, we also control the port power consumption by efficiently consolidating data flow to low-cost switch port. The problem is NP-hard. Thus, Simulated Annealing is used to find near optimal solution of the problem within reasonable computation time.

In Chapter 5, we propose efficient NFV MANO strategies in the cloud computing infrastructure. NFV service is offered by connecting multiple VNFs by service providers policy, which is called a Service Chain (SC). NFV-SC dedicates a set of VNFs having a sequence of processing required for service provider's policy or user requirements. The service providers

force the sequence of VNF processing depending on the operational policy. Since VNFs receive heterogeneous requests from customers depending on service chains, it is important to allocate appropriate resources to VNFs. Also, the service chain embedding in distributed cloud infrastructures affects the performance of the service chain because the service chain needs to forward packets following the processing sequence of service providers. If VNFs in the same service chains are highly distributed across geographically dispersed data centers, it affects the communication time of the service chain. Thus, it will degrade the QoS of the service chain. We study efficient NFV-SC resource allocation and embedding strategies in two components of NFV MANO: 1) VIM and 2) NFVO and VNFM. VIM controls and manages the NFV infrastructure computing, memory, and network resources in a single domain of the infrastructure. Therefore, multiple VIMs exist across the wide area in each domain, and control and manage localized resources depending on NFVO requests. For the localized operation optimization, we propose a strategy which minimizes the processing time of NFV-SC within the given capacity. We employ BCMP mixed queuing network model to analyze the system performance. Our purpose is minimizing the expected waiting time of service chains through optimal service rate allocation for each VNF depending on arrival rates to VNFs. The NFVO manages global resource management, validation, and authorization of NFVI resource requests. The NFVO and VNFM serve as NFV-SC lifecycle management including instantiation, scale-out/in, performance measurement, event correlation across the multiple network domains. Therefore, the role of NFVO is extremely important for the optimal performance of NFV services in cloud computing systems. We develop efficient NFV-SC resource allocation and placement strategy in hyper-scale cloud infrastructure. The purpose of NFVO is minimizing the power consumption of the infrastructure for serving NFV services. We consider the processing time and power in VNFs affected by resource allocation and network, and the network time and power affected by NFV-SC placement. NFV-SC is modeled by an approximate M/G/1/K queuing network and the resource allocation problem is modeled as a Mixed Integer Nonlinear Programming (MINP). Due to the complexity of the global optimization, we propose a heuristic algorithm which uses a correlation clustering, one of the unsupervised machine learning algorithm.

## 1.5 Dissertation Organization

The rest of dissertation is organized as follows. Chapter 2 introduces related literature that study cloud computing in terms of its operation optimization, performance analysis, and modeling strategies. Chapter 3 describes the detailed strategy of optimal data set allocation in distributed heterogeneous clouds. In Chapter 4, we introduce the adaptive data center activation model with user request processing algorithm. Chapter 5 presents the efficient NFV MANO strategy in cloud computing infrastructures. Chapter 6 concludes the dissertation and introduces future work to expand the research achieved of this dissertation.

# CHAPTER2. LITERATURE REVIEW

## 2.1 Virtual Machine Placement in Clouds

Since clouds have heterogeneous resources, it is possible to achieve various objectives by allocating virtual machines according to different strategies. For example, in distributed Clouds, it is important to reduce latency that is mainly affected by the geographical distance between data centers. The model is optimized using one of two objective functions. This could be done by using standard assignment algorithms as in [9]. Alicherry et al. used an integer linear programming model for reducing latency between virtual machines. The first is minimizing the total access time and the other is minimizing the maximum access time. Minimizing total access time can reduce overall access time of running the job. However, minimizing maximum access time is used when performance of the job is the most important. In other words, minimizing the maximum access time could guarantee performance of the application but it was increased the total access time compared to minimizing total access time.

Anothre objective is minimizing power consumption, and many optimization formulations have been introduced to reduce the cost of power consumption in Clouds. Optimizing power allocation and load distribution has been achieved in [15] by using a queuing network model. Also, a load balancing model was introduced to reduce Carbon Emission in [13]. Minimizing the cost of virtual machine usage has not received enough attention even though it is a very important factor from a users perspective. Also, only a few attempts to optimize multiple objectives in Clouds exist in the literature. In [16], Ruiz-Alvarez et al. minimized cost, latency, and bandwidth. However, they have scalarized multi objectives to a single objective function. Consequently, the problem could not guarantee Pareto optimality of each objective.

## 2.2 Power Consumption Minimization in Clouds

Energy saving in data centers is considered in many aspects because of its importance. Junwei Cao *et al.* propose optimal power allocation and load distribution in a heterogeneous data center environment [33]. They model a multicore server processor as a queuing system with multiple servers and prove optimal server setting for two different core speed models, idle speed model and the constant speed model.

Kuangyu Zheng *et al.* propose joint power optimization of data center network and servers with correlation analysis [34]. They propose a power optimization strategy that leverages workload correlation analysis to jointly minimize total power consumption of servers and data center network. After they analyze the correlation between Virtual Machines (VMs), they consolidate VMs that are not correlated with each other onto the same physical servers in order to save server power consumption and consolidate network traffic to save data center network power. Through this approach, they could achieve up to 51.6% of energy saving in their testbed. The proposed approach in this thesis has a similarity with our proposal in Chapter 4 in terms of joint optimization of data center networks and servers. However, the author consolidates VMs based on correlation analysis between VMs, and then activates traffic flow between VMS having correlations, which means the activation of servers and switches are not jointly considered. Also, the traffic flow is consolidated between correlated VMs.

Our research has differences from all previous works in the following aspects: Our prediction model predicts the probability distribution parameter in a certain period instead of predicting quantified values of requests. Therefore, we make a more flexible prediction by deciding the desired probability of predicted distribution. The dynamic data center activation model integrates the dynamic operation of switch layer and host layer while guaranteeing the performance of the data center network. Since we jointly decide activated switches and hosts, we could save more energy compared to other dynamic activation models which determine operating hosts and switches independently.

## 2.3 Queuing Theory in Clouds

Jordi Vilaplana *et al* studied the computer service QoS in cloud computing based on queuing theory [59]. Cloud platforms are modeled by an open Jackson network which consists of multiple nodes, where each node corresponds to a queue in which the service rates are node dependent. They modeled a multi-server system with Entering Server (ES) and Processing Server (PS). ESs are modeled by M/M/1 queues, which represents the load balancer, and PSs are modeled as M/M/m queuing system. However, all queues are assumed to have the same service rate so the heterogeneity of systems is not considered. By using the queuing system analysis, they analyzed the response time of the global cloud architecture with different parameters.

Wei-Hua Bai *et al* investigated the heterogeneous modern data centers and the service process in data centers by using queuing theory [60]. They built a complex queuing model composed of the master server and computing nodes. The master node works as the main scheduler to allocate resources for tasks to dispatch a node to execute tasks, which is modeled as M/M/1/K queue. Computing nodes which are multi-core servers are modeled as M/M/c queues because each multicore execution server can parallel-process multiple tasks. By using queuing theory analysis, they could investigate system metrics such as the mean response time, the mean waiting time, and other important performance indicators. Although they investigated the system performance of heterogeneous data center efficiently, they did not include the switch node in their consideration. Since the switch architecture and connections with servers significantly affect data centers performance, data center switches also need to be considered when analyzing the data center performance in queuing perspective.

Over the system performance analysis, Junwei Cao *et al* studied the problem of optimal multiserver configuration for profit maximization in cloud computing environments [61]. They included the amount of service, the workload of an application environment, the configuration of multiserver system, the service level agreement, the satisfaction of a consumer, the quality of service, the penalty of a low-quality service, the cost of renting, the cost of energy consumption, and the service provider's profit margin in their profit modeling. They maximized their profit through the optimal multi-server configuration. They modeled a multiserver system as

an M/M/m queuing model so that the optimization problem can be formulated and solved analytically. For the more general analysis, they derived the density function of the waiting time of a newly arrived service request and the expected service charge to a service request is calculated.

## 2.4 Traffic Prediction in Clouds

Requests prediction in Cloud and data center has been studied by many researchers for the automatic scaling of systems.

Akindele A. Bankole *et al.* employ machine learning for predictive resource provisioning in Cloud [45]. Their prediction model is achieved with machine learning techniques including: Neural Network, Linear Regression, and Support Vector Machine. They predict the CPU utilization, response time, and throughput based on collected data from virtual machines web servers and database servers. The prediction model generates prediction values in every given minute with machine learning techniques and measure error rate with Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). However, the prediction model did not show a high prediction accuracy, and their results show 24% prediction error in predicting the CPU utilization at a certain point of time and a 21% error in the response time prediction.

Sedeka Islam *et al.* present more advanced machine learning techniques for predicting resource usage in [57]. The error correction Neural Network (ECNN) and the Linear Regression techniques are employed for prediction. They included a sliding window method to reflect the current state of the system. They generated prediction values based on window sizes and evaluated them with MAPE, PRED(25), RMSE, and $R^2$prediction accuracy. The CPU utilization data is collected from the Amazon EC2 cloud through the TPC-W benchmark and prediction values are generated with the ECNN and Linear Regression method. The prediction values of CPU utilization has around 19% error rate without the sliding window and has minimum 18.6% error rate when they employ the sliding window.

Many statistical approaches are also applied to the prediction in clouds. Bruno Lopes Dalmazo *et al.* propose a traffic prediction approach based on a statistical model where observations are weighted with Poisson distribution inside a moving window [44]. They consider

the past information by means of a sliding window of size $\lambda$ and this window is applied by weighting the past observations according to the Poisson distribution with parameter $\lambda$. Dropbox trace data is employed for testing their prediction model and Normalized Mean Square Error (NMSE) evaluation method is utilized for the error measurement. The prediction model could achieve NMSE values between 0.044 and 0.112. The prediction is ideal when NMSE value is equal to zero and worsens when NMSE is greater than one. This approach achieves a reasonably accurate prediction with this approach.

They also propose a traffic prediction model with a dynamic window approach [58]. The sliding window size is changed based on variance of the previous window size. A small variance indicates the predicted data is close to the actual data while a high variance means the predicted data is spread out from the mean. Therefore, they update the window size in every prediction interval by considering the size of the variance in the previous prediction. The prediction accuracy is improved from 7.28% to 495.51% compared to the previous statistical model.

## 2.5 Virtual Network Embedding in Clouds

Virtual Network Embedding (VNE) problems have been studied by many researchers. VNE problems are related to the NFV resource allocation problem in terms of dealing with virtual nodes and virtual links placement onto a substrate network with various objectives. The general, VNE problem has been shown to be NP-hard [94]. Since the problem is intractable, most of the proposals use heuristic solutions.

Juan Felipe Botero *et al* extended the traditional VNE problem to energy awareness and modeled the problem as a Mixed Integer Problem (MIP) [95], [96]. The exact optimal solution of the problem is obtained for small size networks [95]. Then they extended the problem to a scalable energy aware reconfiguration heuristic approach so that the algorithm can be applied to large size networks.

Vincenzo Eramo *et al* proposed reconfiguration cost and energy aware VNE policies in cyclic stationary traffic scenarios [97]. They considered the cyclically static traffic scenario and found an efficient VNE solution over the static time period. The reconfiguration cost is considered in their solution. If the benefit of the reconfiguration of the virtual network is greater than

the current embedding state, the virtual network is migrated to another substrate network. Since the problem is NP-hard, they proposed a heuristic algorithm which use Markov decision process. The numerical results showed they could achieve 35% cost reduction compared to the traditional migration policies.

The embedding cost minimizing virtual node and link mapping algorithm was proposed in [98]. The authors included the geographical location of the substrate and virtual nodes in the problem's constraints and considered the influence of the geographical distance between nodes in the embedding strategy. The problem is formulated as an MIP and is solved using a relaxation strategy with two approaches: deterministic and random.

VNE was also studied in different perspectives. A security-aware VNE strategy is proposed in [99], [100]. The approach described in [101] proposed a way to increase the survivability of virtual nodes by introducing link failures in VNE. Also, the authors in [102] treated three types of failures: virtual node failure, physical node failure, and physical link failure.

As an extension of the traditional VNE embedding problems, many strategies are proposed in NFV resource allocation. A heuristic approach proposed in [103] solves service chain composition and VNF Forwarding Graph (FG) embedding problem in a coordinated way. The VNF-FG is composed of a service chain that fulfills the service requirement and VNF-FG embedding strategy is searched while it computes the result within a reasonable runtime with negligible performance effects.

### 2.6 Network Function Virtualization Deployment in Clouds

The VNF placement problem has been studied widely because of its influence on system cost and efficiency. A VNF placement model is proposed by applying NFV and SDN to LTE mobile core gateways in [67].

In [68], virtual Deep Pack at Inspection (vDPI) function placement problem is studied. Mathieu Bouet *et al.* formulated the problem as an Integer Linear Program (ILP) and minimized the management and operation costs.

Sevil Mehraghdam *et al.* suggested a model formalizing the chaining of VNFs using a context-free language and proposed a mapping scheme to the network by using Mixed Integer

Quadratically Constrained Program (MIQCP) for finding the optimal placement in [69].

The VNF placement problem is also studied in the data center and cloud environments. Ming Xia *et al*. identified the possibility of minimizing expensive optical/electronic/optical conversions for NFV chaining in data centers. They formulated the problem of optimal VNF placement as a Binary Integer Program (BIP) and proposed a heuristic approach to solve the problem in [70].

Queuing network theory is studied by many researchers. However, the resource allocation problem in queuing network models has not been studied well enough. Optimal server resource allocation problem is presented in [71] only for open queuing network model. Alex Zhang *et al*. suggested nonlinear integer programming model for determining the number of machines in a multi-tier server network. They used an open queuing network model average response time to minimize the total number of machines while satisfying the average waiting time of open queuing network.

The service rate allocation problem in queuing systems is studied in [72] by Onno J. Boxma. He formulated the model that allocates the optimal service rates in Jackson networks and provided closed form expressions for optimal service rate allocation to all stations in the network.

Jocelyne Elias *et al* studied congestion mitigation network function composition problems as a non-linear optimization model [104]. They used a game theoritic approach to minimize the congestion of the physical infrastructure when all virtual operators independently select their network. They demonstrated that the NFV congestion minimization game has a unique Nash Equilibrium under general conditions, and the solution can be easily computed.

The game theoritic approach is also employed by other researchers in the NFV-RA problem [105]. Roberto Bruschi *et al* presented an energy-aware game theory based solution for resource allocation of VNFs within NFV environment. They considered each VNF as a player who competes for substrate node while minimizing the individual energy cost.

Sijia Gu *et al* also proposed an efficient auction mechanism solving the resource allocation problem as an NFV while pursuing the maximization of social welfare [106]. They formulated the problem by Integer Linear Program (ILP) which is NP-hard. They proposed an efficient primal-dual approximation algorithm to compute a close to optimal resource allocation solution.

## 2.7 Chapter Summary

This chapter introduces related works in cloud computing system modeling, analysis, and operations perspective. Although many research and contributions are achieved by many researchers, this dissertation has an originality compared to previous works. In utilizing queuing theory, this dissertation uses various practical queuing model. For example, M/G/1/K queuing network model is used to NFV-SC modeling by considering a high level of service time variation in NFV environment. Also, we used M/M/1 queuing model for data center switch and M/M/c queuing model for data center server. This helps to analyze the performance of the data center more practically.

In data center power saving, this dissertation proposes joint adaptive activation of data center switch and server layers. The proposed model efficiently minimizes the power consumption of the data center while guaranteeing QoS and connectivity.

# CHAPTER3. OPTIMAL DATA SET ALLOCATION IN DISTRIBUTED HETEROGENEOUS CLOUDS

This Chapter introduces the optimal data set allocation strategy in heterogeneous cloud infrastructures. We formulate a problem as multi-objective linear programming model which minimizes the processing time and cost and propose a Pareto front searching algorithm.

## 3.1 Problem Formulation

Our goal is to optimize the allocation of data sets to cloud computing systems that have heterogeneous resources. We have two objective functions: 1) processing time and 2) cost. Processing time minimization and cost minimization are contradictory. In order to decrease processing time, it is helpful to allocate data sets to high performance virtual machines. However, data set placement to high performance virtual machines increases the cost. Therefore, we attempt to find Pareto optimal points that guarantee optimality of processing time and cost simultaneously.

Large size data sets will be divided into small data set, and the data sets will be assigned to different virtual machines that have different resources. We assume parallel processing of data sets and do not consider serial processing of partitioned data sets.

We develop a linear programming model for this problem. There are two objective functions that have the data set size as a decision variable. Both objective functions share the same constraints. There are four types of constraints in the problem: Storage capacity, summation of total data set size, individual data set size, and communication latency. Table 1 shows the glossary of all terms used in our model.

Table 3.1: Formulation Notations

| Variable | Description |
|---|---|
| $D_{mnl}$ | Continuous variable, the dataset size allocated to the $l$th machine in the $n$th data center, which is derived from $m$th dataset. |
| $i$ | Constant, the number of datasets. |
| $j$ | Constant, the number of data centers. |
| $K_n$ | Constant, the number of virtual machines in $j$th data center. |
| $DS_{mn'}$ | The $m$th dataset initially submitted to $n$th data center. |
| $B_{nl}$ | Constant, network bandwidth of $n$th data center $l$th virtual machines(GB/sec). |
| $B_{n'n}$ | Constant, bandwidth for data transfer from $n'$th data center(initially arrived) to $n$th data center (GB/sec). |
| $P_{nl}$ | Constant, Processing time of $n$th data center $j$th virtual machine (GB/sec). |
| $CC_{nl}$ | Constant, communication cost for uotgoing data from $n$th data center $l$th virtual machine ($/GB). |
| $TC_{j'n}$ | Constant, transfer cost for outgoing data from initially arrived data center $j$ to target data center $n$ (GB/sec). |
| $VC_{nl}$ | Constant, virtual machine cost of data center $n$th $l$th virtual machine ($/GB). |
| $C_{nl}$ | Constant, available capacity of $j$th virtual machine in data center $n$. |
| $LT_{nl}$ | Constant, maximum available latency of communication time from $n$th data center (sec). |
| $BC_{nl}$ | Constant, cost to upgrade the bandwidth of $n$th data center to virtual machine $l$ ($). |
| $\alpha$ | Constant, communication overhead rate from original dataset size ($0 \leq \alpha \leq 1$). |

### 3.1.1 Processing Time

$$\underset{D_{mnl}}{\text{minimize}} \sum_{m=1}^{i} [\sum_{n=1}^{j} \sum_{l=1}^{kn} \frac{\alpha}{B_{nl}} D_{mnl} + \sum_{\substack{n=1 \\ n \neq j'}}^{j} \sum_{l=1}^{kn} (1+\alpha) D_{mnl} + \sum_{n=1}^{j} \sum_{l=1}^{kn} \frac{D_{mmnl}}{P_{nl}}] \tag{3.1}$$

The processing time objective function includes time to allocate sub-data sets to other data centers, virtual machine processing time, and time to send data set overhead to cloud manager for share their processing status. Each data center has a cloud manager. The data center that receives data set first becomes cloud manager for the data set. Therefore, we

consider data transfer time from initial arrived data center and communication time to send processing status to initial data center, and virtual machine processing time of sub-data sets in each virtual machine. The first term is communication time between all virtual machines. Since a large size data set is divided into several smaller data sets, each virtual machine needs to check processing status of sub-data sets. Therefore, after finishing the processing in each virtual machine, virtual machines send their status to the cloud manager. We consider only the outgoing time from allocated virtual machines for communication time. Some cloud service providers provide high bandwidth for outgoing data transfer from the virtual machine with additional cost. Thus, only considering outgoing time from the virtual machine will be the dominant factor in communication time. We do not need to transfer all data for communication. Communication overhead is configured with some rate from data set size $D_{mnl}$, allocated sub-data set in the $l$th virtual machine in nth data center. By dividing the data set overhead size with bandwidth of $l$th virtual machine in nth data center, communication time of the virtual machine can be obtained.

The second term refers to the data transfer time. The initial data set is such that their required resource exceeds available resources in a certain data center. If it is able to reduce processing time by dividing initial data set and allocating sub-data sets to other virtual machines in a certain data center, we have to transfer the data sets to other data centers virtual machines. This process would consume some transfer time. So transfer time can be calculated by dividing allocated data set and communication overhead size by communication bandwidth of the initially allocated data center. We assume data sets have no serial relationship. All data sets are possible to be processed in parallel model in this problem.

The third term refers to the virtual machine processing time. Virtual machine processing time is one of the most important factors in deciding the processing time objective value. Cloud service providers offer different types of service depending on users demands. The most representative difference is the number of CPUs. High performance virtual machines provide more number of cores. So it is able to reduce processing time by the allocating more data to the virtual machine that has high performance processor or high processing ability.

### 3.1.2 Cost

$$\underset{D_{mnl}}{\text{minimize}} \sum_{m=1}^{i}[\sum_{n=1}^{j}\sum_{l=1}^{kn}\alpha D_{mnl}CC_{nl} + \sum_{\substack{n=1 \\ n\neq j'}}^{j}\sum_{l=1}^{kn}(1+\alpha)D_{mnl}TC_{j'n} + \sum_{n=1}^{j}\sum_{l=1}^{kn}\frac{D_{mnl}}{P_{nl}}VC_{nl}] + \sum_{n=1}^{j}\sum_{l=1}^{kn}BC_{nl} \tag{3.2}$$

We consider similar factors in the cost objective function: communication cost, data transfer cost, virtual machine cost, and bandwidth cost. The first term stands for communication cost. Most cloud computing service providers charge for only outgoing data transfer depending on data size and communication bandwidth. In the first term, we only consider cost for data set size. Similar to communication time, only outgoing communication overhead is charged for communication cost.

The second term is the data transfer cost. As mentioned in section 3.1.1, initial data set has to be moved to other data center as necessary. Since data transfer cost is also only charged for outgoing data, we multiply cost by data set and overhead size.

The third term is virtual machine cost. Virtual machine cost charged depends on hours of use and its performance. High performance virtual machines are more expensive than low performance virtual machine. It may seem cheaper if we just use low performance virtual machine to reduce virtual machine cost, but virtual machine cost also increases with usage time of the virtual machine. If the virtual machine processing time increases, the virtual machine cost is increased simultaneously. So the virtual machine cost is multiplied by virtual machine processing time in the third term in order to make the best decision between performance and usage time of virtual machine.

Bandwidth cost is not proportional to data set size. When user wants to upgrade communication bandwidth of certain virtual machines, most cloud service providers charge additional cost per month. We do not consider that long term processing time over a month. Therefore, bandwidth cost is fixed and is not affected by data set allocation.

### 3.1.3 Constraints

$$D_{mnl} \leq C_{nl}, \forall m, n, l \tag{3.3}$$

$$\sum_{n=1}^{j} \sum_{l=1}^{kn} D_{mnl} \geq DS_{mn'}, \forall m \tag{3.4}$$

$$D_{mnl} \leq DS_{mj'} \tag{3.5}$$

$$\sum_{l=1}^{kn} \frac{\alpha D_{mnl}}{B_{nl}} \leq LT_n, \forall m, n \tag{3.6}$$

(3.3) is virtual machine capacity constraint. Virtual machines have their own given amount of storage. New allocated data set size cannot exceed the current available storage size of virtual machine.

(3.4) is summation of allocated data set size constraint. Total size of assigned data sets originated from same data set has to be greater than or equal to the original data set size.

(3.5) is each data set size constraint. Allocated data set size cannot be greater than original data set size.

(3.6) is latency constraint. This provides performance guarantees on the communication time from each data center. Latency will be different according to the types of data sets. Some data sets are sensitive to latency but some are not. Therefore, we can set latency constraint depending on data sets property.

The problem is multi-objective linear programming model. There are many known algorithms for solving single objective linear programming model like Simplex algorithm, Criss-cross algorithm, and Conic sampling algorithm. After solving each objective function, we will use a special case Bensons algorithm to find Pareto front of optimal solutions.

## 3.2 Algorithm for Multi Objective Optimization

In this section, we introduce a heuristic algorithm for finding the Pareto front of the multi objective linear programming model. The algorithm, which is shown in Algorithm 1, is designed

for the maximization problem. We will reformulate the problem to an equivalent vector form of the maximization problem.

$$\begin{aligned} \underset{D_{mnl}}{\text{minimize}} \quad & Zeta_p = -P^\top D \\ & Zeta_c = -C^\top D + b \end{aligned} \tag{3.7}$$

$P^\top$ denotes the transpose of processing time coefficient vector about each $D$. $D$ denotes the vector of decision variables $D_{mnl}$. Since $P_\top$ and $D$ have the same dimension, we can obtain the objective value of processing time. Let us assume we allocate one dataset to two data centers which have 2 virtual machines each. Then $D$ vector becomes $[D_{111}\ D_{112}\ D_{121}\ D_{122}]$. The elements of vector $P$ are summation of each term in (3.1). The each element in $P$ will be summations of the data transfer time, virtual machine processing time, and communication time about all $D_{mnl}$ elements in the. $CT$ is also transpose of cost coefficient vector having same dimension with $D$. Elements of the $C$ vector are the summations of each term in (3.2) for all elements of $D$, just as like the $P$ vector. $b$ is the summation of all bandwidth cost, the last term in (3.2), which is not relevant to the decision variable $D_{mnl}$. So we can also obtain objective function of cost.

The algorithm starts with two Pareto optimal solutions of each objective functions: $D_p$ and $D_c$. If $D_p$ and $D_c$ are on the same line, we can say all points between $D_p$ and $D_c$ are Pareto solutions. In order to verify if the $D_p$ and $D_c$ are on the same line or not, a new objective function is generated that has potential optimal points between $D_p$ and $D_c$.

$$Zeta_k = [P^\top (D_c - D_p)C + C^\top (D_p - D_c)P]^{top} D \tag{3.8}$$

(3.8) is a new objective function that has optimal values between the optimal values of $D_p$ and $D_c$. The first term in (3.8) is processing time vector having size between processing time of $D_p$ and $D_c$ in processing time. Also, the second term in (3.8) is cost vector having size between cost of $D_c$ and $D_p$ in cost axis.

By summing two vectors, we can get new vector having new Pareto optimal point not dominated by both of $D_p$ and $D_c$. Since we solve new objective function by same constraints,

we can obtain new feasible optimal solution Dk. The next step is observing whether three Pareto points are in the same line or not. If $Zeta_k(D_k) = Zeta_k(D_p) = Zeta_k(D_c)$, the three points are in the same line. If they are on the same line, which guarantees that the line between $D_p$ and $D_c$ is the Pareto front. If they are not on same line, we have to find new optimal points between $D_p$ and $D_k$, and then between $D_k$ and $D_c$. If we can find all pairs of adjacent lines connecting all Pareto points, we stop the iteration of the algorithm.

**Algorithm 1** Exploring Pareto Front Algorithm

**INPUT**: $D_p$, $D_c$, $Zets_p(D_p)$, $Zets_c(D_p)$, $Zets_p(D_c)$, $Zets_c(D_c)$
$D_i = D_p$, $D_j = D_c$
Hash($K_l$, $D_i = D_p$, $Dj = Dc$, Optimal pairs)
$l = 1$
**while** **do**
  $K_l = \frac{Zeta_p(K_l,D_i)+Zeta_c(K_l.D_i)+Zeta_p(K_l.D_j)+Zeta_c(K_l.D_j)}{2}$
  $K_{list} \leftarrow K_l$
  $Zeta_{K_l} = [P^\top(K_l.D_i - K_l.D_j)C + C^\top(K_l.D_j - K_l.D_i)P]^\top D$
  $D_{kl} = Findoptimalsolutionfor Zeta_{K_l}$
  **if** $Zeta_{K_l}(D_i) = Zeta_{K_l}(D_{kl}) = Zeta_{K_l}(D_j)$ **then**
    Line between $D_i$ and $D_j$ are Pareto Front
    $K_l \leftarrow$ Optimal Pair ($D_i$, $D_j$)
    **if** $L$ is the largest index among $K_l$ in $K_{list}$ **then**
      break;
    **else**
      $l = l + 1$
    **end if**
  **else if** $Zeta_{K_l}(D_i) = Zeta_{K_l}(D_{kl}) \neq Zeta_{K_l}(D_j)$ **then**
    Line between $D_i$ and $D_{kl}$ is Pareto Front
    $K_l \leftarrow$ Optimal Pair ($D_i$, $D_{kl}$)
    Make hash
    Hash($K_{l+1} = (D_i, D_{kl})$, $D_i = D_i$, $D_j = D_{kl}$ optimal pair)
    $l = l + 1$
  **else if** $Zeta_{K_l}(D_i) \neq Zeta_{K_l}(D_{kl}) = Zeta_{K_l}(D_j)$ **then**
    Line between $D_j$ and $D_{kl}$ is Pareto Front
    $K_l \leftarrow$ Optimal Pair ($D_{kl}$, $D_j$)
    Make hash
    Hash($K_{l+1} = (D_i, D_j)$, $D_i = D_{kl}$, $D_j = D_j$, optimal pair)
    $l = l + 1$
  **else if** $Zeta_{K_l}(D_i) \neq Zeta_{K_l}(D_{kl}) \neq Zeta_{K_l}(D_j)$ **then**
    Make hash
    Hash($K_{l+1} = (D_i, D_j)$, $D_i = D_i$, $D_j = D_{kl}$, optimal pair)
    Hash($K_{l+2} = (D_i, D_j)$, $D_i = D_{kl}$, $D_j = D_j$, optimal pair)
  **end if**
**end while**
Sort $K_l$ in $K_{list}$ by $K_l$ value
$K'_{list}$: Sorted $K_l$ list
**for** $n = 1$; $n = l$; $n$++ **do**
  Obtain optimal value pairs of all Pareto points in optimal value domain
  $O_n$=($Zeta_p(K'_n.D_i)$, $Zeta_c(K'_n.D_i)$), ($Zeta_p(K'_n.D_j)$, $Zeta_c(K'_n.D_j)$)
**end for**
If we connect all pairs of $O_n$, we can get Pareto Front.
**OUTPUT**: All pairs of Pareto points and Pareto front

The algorithm starts from two optimal points of each of the objective functions and objective values (Line 1). In order to manage each point efficiently, hashmap function is employed including key value, two nearest optimal points not on the same line, and optimal pairs when optimal pairs exists (Line 4). Key value, which shows position of points, is determined depending on optimal values of nearest two points. By averaging optimal values of nearest two points, we can get key value sequentially to arrange optimal pairs (Line 8). After obtaining key value, Kl is saved in Klist (Line 9). New objective function is generated with two optimal points in Kl hashmap as explained in previous paragraph, and then solves the problem with the same constraints (Line 10  Line 11). If optimal value of $Zeta_{kl}(D_i) = Zeta_{kl}(D_{kl}) = Zeta_{kl}(D_j)$, three points are in the same line. So we add optimal points $D_i$ and $D_j$ to $K_l$ hashmap as optimal pairs. If we cannot find anymore $K_l$ in the $K_{list}$, then the Pareto front has been found. So we stop the iteration (Line 12  Line 21).

If $Zeta_{kl}(D_i) = Zeta_{kl}(D_{kl}) \neq Zeta_{kl}(D_j)$, $D_i$ and $D_{kl}$ are in the same line but $D_{kl}$ and $D_j$ are not in the same line. Therefore, only $D_i$ and $D_{kl}$ are added to $K_l$ hashmap as optimal pairs. Since $D_{kl}$ and $D_j$ are not in the same line, new objective function having optimal point between $D_{kl}$ and $D_j$ has to be generated again. So we make new hashmap $K_l$+1 including two points $D_{kl}$ and $D_j$ and empty optimal pairs (Line 23  Line 30). If $Zeta_{kl}(D_i) \neq Zeta_{kl}(D_{kl}) = Zeta_{kl}(D_j)$, $D_{kl}$ and $D_j$ are in the same line but $D_i$ and $D_{kl}$ are not in the same line. As like previous paragraph, we save $D_{kl}$ and $D_j$ to optimal pairs of $K_l$ hashmap and make new $K_l + 1$ hashmap including $D_i$ and $D_{kl}$ (Line 31- Line 38).

If all three points are not in the same line, two hashmaps are generated including $D_i$ and $D_{kl}$, and then $D_{kl}$ and $D_j$ (Line 39  Line 45).

The iteration is repeated until there is no more $K_l$ in the $K_{list}$. If we cannot find any more $K_l$, this guarantees that every hashmap $k$ will include optimal pairs or empty pairs. By sorting $K_l$ according to the $K_l$ key value, $K_l$ will be ordered sequentially. Then the Pareto front can be drawn by connecting all optimal points in $K_l$ (Line 49  Line 54).

## 3.3 Simulation Model

In this section, we evaluate the performance of the model and find Pareto optimal points of the multi-objective problem by using the proposed algorithm. We consider a cloud consisting of 10 data centers having heterogeneous virtual machines. Data center resource information is configured from actual data of cloud computing service providers. Five types of cloud data center infrastructures are employed: Microsoft Azure, Rackspace, Google cloud, Amazon EC2, and IBM cloud. Each cloud provider offers different infrastructure services. For an example, Microsoft Azure has four options for CPU and has an option for upgrading network bandwidth. Rackspace provides all different network bandwidth depending on performance of virtual machines. A single core virtual machine provides 60 Mbps of bandwidth, but a dual core virtual machine has 120 Mbps network bandwidth. Google cloud provides the highest performance virtual machine. Google services provide 16 core virtual machine model. All data centers have heterogeneous resources similar to this. Microsoft Azure (data centers number 1 and 6) includes 8 types of virtual machines, Rackspace(data centers number 2 and 7) has 10 types of virtual machines, Google cloud (data centers number 3 and 8) has 5 types of virtual machines, Amazon EC2 (data center number 4 and 9) has 12 types of virtual machines, and IBM cloud (data center number 5 and 10) includes 8 types of virtual machines, and two data centers exist from each data center platform. So we have a total of 86 types of virtual machines to allocate data set. Each virtual machine in a different data center has a different cost, processing time, and bandwidth. We generate the storage capacity of virtual machine randomly between 0 to data the storage size of virtual machines for constraint (3). Three data sets are allocated to the cloud. The first data set has 1000 GB size and assigned to data center 3. The second data set is assigned to data center 1 with 500 GB size. The third data set is placed to data center 4 with 700 GB size.

## 3.4 Simulation Result

We used GUSEK, open source Linear/Mixed Integer Linear Programming solver to solve the two objective functions separately. The problem includes 258 decision variables and 548

constraints. In order to find the Pareto front, we implemented Algorithm 1 in MATLAB. Figure 1 is the result of minimizing processing time. $D_{nl}$ stands for the $l$th virtual machine in $n$th data center. High performance virtual machine has large index number $l$.

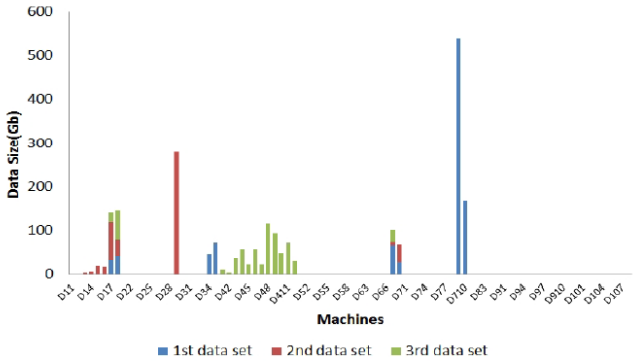


Figure 3.1: Minimum processing time optimization result

In Fig. 3.1, we can see most of the data set is assigned to high performance virtual machines. Also, the data set is mainly assigned to the initial data center to which it is submitted in order to reduce the data transfer time. The third data set shows that 83% of the data set is assigned to data center 4 in order to reduce data transfer time. Also, 90% of the first data set is allocated to high performance virtual machine of data center 9 and 10 in the result.

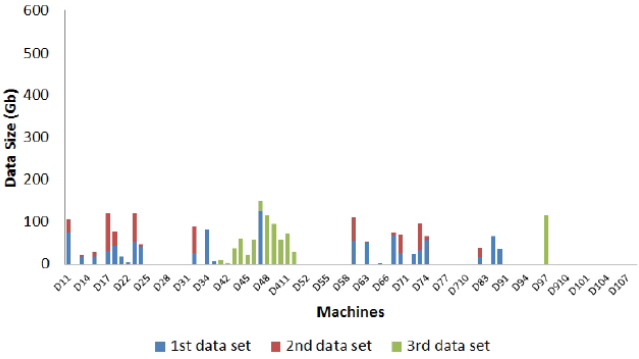


Figure 3.2: Minimum cost optimization result

Fig. 3.2 is result of minimizing the Cost objective function. Compared to processing time minimization problem, the data sets are evenly distributed. Cost minimization also shows a similar allocation trend. In order to reduce data transfer cost, the data set is mainly allocated to the initial data center. However, most data sets are not only assigned to low performance

virtual machines, but many data sets are also allocated to high performance virtual machines. This means that virtual machine cost is not a dominant factor in deciding the Cost. It seems like the effect of the data transfer cost and communication cost is relatively more important than virtual machine cost according to the result.

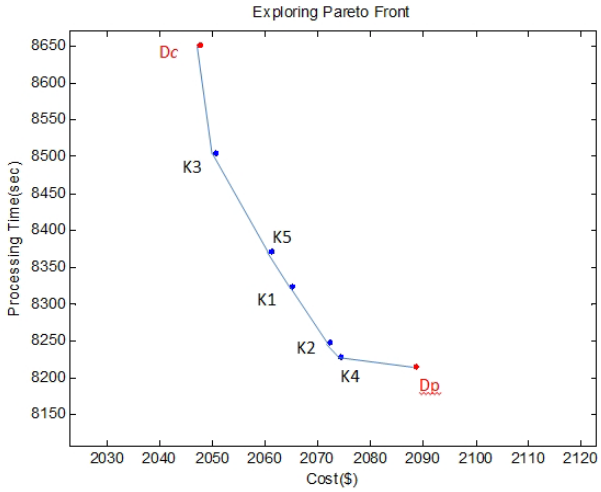


Figure 3.3: Pareto optimal points with iterations

According to the optimization result of two separate objective functions, we could find two Pareto optimal points. The first point has cost value of 2088 and processing time of 8215. The second point has cost value of 2047 and processing time value of 8651 in Fig. 3.3. Both points are not dominated by each other. Thus, both points are Pareto optimal points.

According to the introduced algorithm, we could find more Pareto optimal points between initial optimal points.

At iteration 2, we could find two Pareto optimal pairs, $K_1$ and $K_2$. According to the algorithm, $K_1$ and $K_2$ are on the same line but $K_2$ and $D_p$ are not in the same line. So another Pareto optimal points will exists between $K_2$ and $D_p$. Since $K_1$ and $K_2$ are on the same line, we can get partial Pareto front by connecting these two points. In iteration 3, we could find $K_3$ point on the same line with $D_c$. In the same way, the line between $D_c$ and $K_3$ will be partial a Pareto front and there will be one more Pareto points between $K_3$ and $K_1$. After five iterations of the algorithm we could search Pareto points between $D_c$ and $D_p$. If we connect these points sequentially, we can obtain Pareto front of this simulation model.

## 3.5 Chapter Summary

In this Chapter, we have developed a strategy for allocating data sets to VMs in a heterogeneous cloud consisting of different types of data centers with each data center offering different types of VMs. The strategy is based on jointly considering the cost and processing time. We therefore formulated the assignment problem as a dual objective optimization problem, and introduced an algorithm for finding the Pareto front of optimal solutions. The result of separate objective function optimization shows proper characteristics of each objective. When we minimize processing time, the data set is usually assigned to high performance virtual machines. However, data set is allocated relatively to lower performance virtual machine when cost is optimized. With the joint optimization, many of the Pareto points enable us to allocate data sets in many ways depending on properties of data sets.

# CHAPTER4. ADAPTIVE DATA CENTER ACTIVATION WITH USER REQUEST PREDICTION LEARNING ALGORITHM

This chapter proposes an adaptive data center activation strategy with user traffic prediction for power consumption minimization. We propose a traffic prediction algorithms by using a statistical learning approach and adjust activation of the data center depending on the expected traffic.

## 4.1 System Model

### 4.1.1 Distribution Parameter Adaptive Data Center Activation Model

The cognitive cycle of the adaptive data center activation model, which is shown in Fig. 4.1, is composed of three phases: data collection, request prediction, and data center activation. For the first phase, the control plane collects the number of incoming tasks and refines collected data to utilize it for the prediction. The collected data is saved in the prediction data set and the prediction model employs it to forecast future requests of users by using a cyclic window learning algorithm. According to the predicted requests, the control plane solves adaptive activation problem and activates an optimal set of switches and hosts, and keep unnecessary components in the idle state. This cycle is repeated periodically in every predetermined period. For example, if we set the duration to 30 minutes, the system repeats one cycle every 30 minutes. Thus, the system can reconfigure the data center every 30 minutes in order to reduce the waste of resources.

In adaptive data center activation model, the system setting time is not a minor issue in terms of system delay. The inactivated devices keep in standby or low power idle states as described in [62], [63]. If the devices are under the low power idle state for power saving

purpose, a longer system setting time will be required when the device is required to wake up. By using the cognitive cycle of the system, the wake up delay of devices can be minimized through predictive system activation. The prediction process is executed in advance of the system activation transition. Based on the prediction result, the controller can prepare newly activated devices in the standby state. So the system wake up time will be minimized according to prediction and activation schemes. Thus, the system setup time is not considered in our delay constraint.

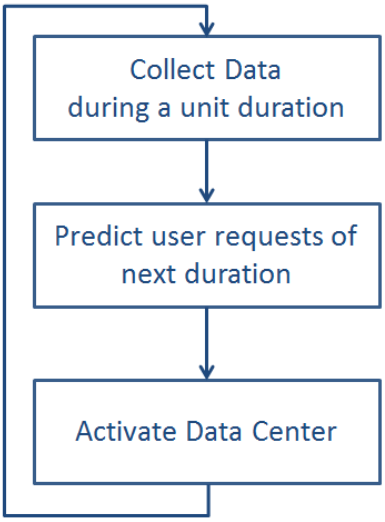


Figure 4.1: Cognitive cycle of system

### 4.1.2 Fat-tree Data Center

There are many data center architectures: Fat-tree, VL2, Bcube, and so on. In this Chapter, we adopt the Fat-tree architecture, the most widely used data center architecture by many Cloud service providers [29], [52].

Fat-tree is composed of three switch layers (core, aggregation, and ToR layer) and a host layer as shown in Fig. 4.2. All switches have the same number of ports. If we assume the number of ports in each switch is $k$, we have $k$ core switches and $k$ Point of Delivery (POD). Core switches are grouped into $k/2$ groups. Since a core switch has $k$ ports, each core switch is connected to all PODs. For example, in Fig. 4.2, the first core switch in

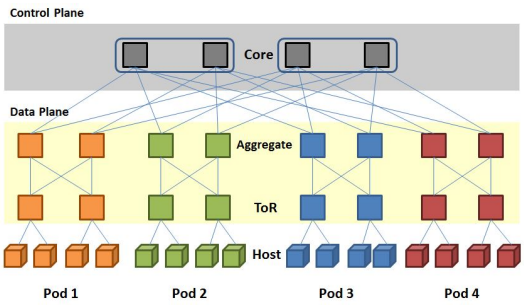


Figure 4.2: SDN fat-tree data center model (k=4)

each group is connected to the first aggregation switch in each POD and the second core switch in each group is connected to the second aggregation switch in each POD. Each POD has $k/2$ aggregation switches and $k/2$ ToR switches. Aggregation, ToR switches, and their interconnection network forms a complete bipartite graph. Since a ToR switch uses $k/2$ ports for connecting to aggregation switches, they can serve $k/2$ hosts each. Therefore, the data center can has a total of $k^3/4$ hosts.

The Fat-tree data center architecture is widely used because of its scalability and interconnection capability [29]. Due to the scalable characteristic of Fat-tree data center, it is able to configure large size data center with lower Capital Expenditure (CAPEX). Also, interconnections between hosts in the same data centers are easy with full bandwidth. In other words, service providers can reduce the expense to build large scale data centers and support high speed communication between servers in the same data center with Fat-tree data center architecture.

In a Fat-tree architecture data center, we cannot deactivate necessary switches to operate required hosts because of its connectivity. For example, the first ToR switch cannot be turned off when we want to allocate jobs to the first host in the first POD. In our model, the core switch should be able to reach any host using three links and a host should be able to reach a core switch using three links as well. Therefore, we will turn on and off switches without violation of this rule to guarantee connectivity.

### 4.1.3 SDN Based Data Center

For comprehensive management of data center network management, we employ an SDN data center model. The SDN data center model is composed of the control plane and the data plane. The control plane plays the role of managing the network and deciding routing paths. The data plane just forwards data according to the values in the forwarding table.

The control plane should have access to data plane switches and needs to receive system status information from all data plane switches. All switches implement the data plane functionality since they have to forward data. In addition, the core switch layer implements the control plane. Since the workload distribution and resource allocation are determined by the control plane switches, the data center can activate only a limited number of switches and hosts by allocating workloads optimally. In traditional data centers, a host could reallocate datasets to other hosts in the same POD by going through just ToR and aggregation switches. However, inter-traffic loads from hosts always go through core switches in the SDN data center model, and improve the overall traffic management of data center.

## 4.2 Request prediction Algorithm

The prediction model estimates parameters of the probability distribution of future user requests in every predetermined period. For the traffic analysis, one week traffic data is selected from Google Cluster trace data in [48]. Since the data is selected starting from a random point of the whole data, we do not know the exact data and time of the traffic measurement. However, we could find that user requests have obvious patterns and the patterns are repeated periodically, as shown in the Fig. 4.3.

The hourly pattern shows high peaks for the first 4 hours and the last 10 hours. So, we can assume daytime starts around the 10th hour from the measurement and end at the 4th hour in Fig. 4.3. If we observe the daily pattern, the first two days and the last two days show higher requests. In the same way, we can assume weekdays start on the 4th day of the measurement by assuming requests increases during weekdays rather than the weekend. The daily pattern analysis of task arrivals presents obvious patterns of the incoming requests. Therefore, the

cyclic data collection should be an effective approach for the prediction. Based on our analysis, the prediction model adopts periodic history data at the same time point in order to make predictions about future data center loads.

We introduce three time scales for the prediction periods: Pattern Period ($PP$), Target Period ($TP$) and Utilization Period ($UP$). The $PP$ is a cyclic interval that exhibits pattern repetition. The $TP$ is a unit duration for which we want to make a prediction. The $UP$ is a cyclic window that we use for predicting the activities in $TP$. In the example, we predict the request distribution during a certain Monday by assuming the same pattern is repeated every week. Then, we can set the $TP$ to a day and the $PP$ to a week because we assume the pattern of a day is repeated every week. If we only use the past Mondays$'$ data for the prediction, the $UP$ becomes one day.

Since patterns are repeated on every $PP$, any time duration for the prediction corresponds to a certain $TP$ on the $PP$. Therefore, we can predict the request distribution during any time interval by correlating them to a certain $TP$ on the $PP$. For precise prediction, data is accumulated for several $PP$s. Although patterns of the traffic distributions will be similar at the same time point in every $PP$, the traffic amount will be different. In other words, we can say the distribution of the traffic shows similar pattern in every Monday, but we cannot ensure that the amounts of traffic will be the same. Therefore, the prediction model can achieve higher prediction accuracy by accumulating data during several $PP$s.

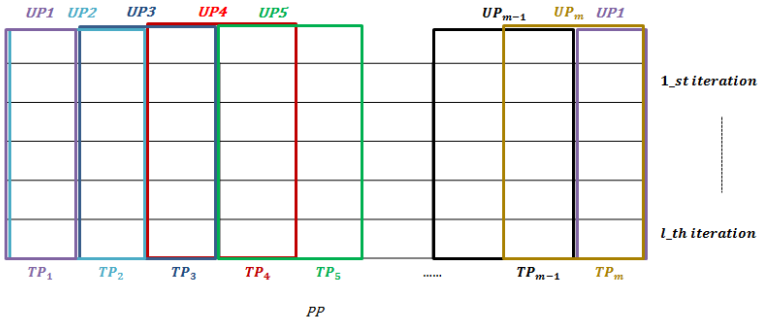


Figure 4.3: Prediction dataset

To implement prediction, the data set saves the past data in an $m \times l$ matrix. $m$ represents the number of $TP$s on the $PP$ and $l$ denotes the number of $PP$s we accumulate. In Fig. 4.4,

each vertical block corresponds to saved parameters of the probability distribution in each $TP$ during a $PP$. We start to stack the data from the first block of the first iteration. If the data set is filled until the $TP_m$'s block, which is the last TP, we move to the second iteration and stack the data from the first block of the second iteration, which means we have saved data during a $PP$. When the matrix is full, the data set goes back to the first block of the first iteration and replaces the old data to reflect the tendency of recent requests.

Any time duration that we want to predict the traffic distribution can be related to a certain $TP$ on the matrix. In order to make a prediction, the $UP$ data is employed. In Fig. 4.4, we can see that distribution parameters of $TP_m$ can be predicted by using $UP_m$. We can set the size of $UP$ depending on how many previous $TP$s will affect the state of the current $TP$. For example, $UP$ is set to two days and $TP$ is one day, e.g., previous Sunday and Monday′s history parameters are utilized to predict next Monday's request distribution.

There are three types of user requests that we want to predict: the number of arriving tasks, CPU requests, and Memory requests. First, the prediction model will construct a histogram of user requests in every target period to observe distributions of requests. Then, it is able to see what kind of probability distribution model is fit to the distribution of requests. When the probability distribution model is decided for fitting, we will adopt MLE in order to obtain parameters of the probability distribution in every observed period and parameters will be saved on dataset. After the dataset accumulates enough data for the prediction, it is able to predict parameters of a following target period. LLR will be employed to predict parameters of the future requests.

### 4.2.1 Histogram Analysis

A histogram is the graphical representation of the distribution of data. We can observe frequencies and overall distribution of given data through a graphical representation.

Histograms of requests are constructed in every regular interval to observe the distribution of requests. After observe the histograms, the prediction model decides which of the probability distribution model will be the closest to the actual distribution of requests.

### 4.2.2 Maximum Likelihood Estimation (MLE)

the Poisson distribution based on the histogram observation of the experiment data. Depending on accumulated request data, the prediction model will induce the Poisson distribution parameter by using MLE in every $TP$.

Poisson distribution has the only parameter $\lambda$. Since the prediction model has data through observation, the number of task arrivals, CPU, and memory request, it is able to induce parameter $\lambda$ by using MLE method. If we observe n independent datasets $X1$, $X2$, $X3$, ..., $Xn$ *iid* Poisson random variables, maximum likelihood function L($\lambda$) will be:

$$L(\lambda) = \frac{\lambda^{X_1} e^{-\lambda}}{X_1} \frac{\lambda^{X_2} e^{-\lambda}}{X_2} \cdots \frac{\lambda^{X_n} e^{-\lambda}}{X_n} = \prod_{i=1}^{n} \frac{\lambda^{X_i} e^{-\lambda}}{X_i} \tag{4.1}$$

If we take log in the equation, log likelihood function becomes:

$$\begin{aligned} l(\lambda) &= \sum_{i=1}^{n} (X_i \log \lambda - \lambda - \log X_i!) \\ &= log\lambda \sum_{i=1}^{n} X_i - n\lambda - \sum_{i=1}^{n} \log X_i! \end{aligned} \tag{4.2}$$

We find maximum of $\lambda$ by finding the derivative of equation:

$$l'(\lambda) = \frac{1}{\lambda} \sum_{i=1}^{n} X_i - n = 0 \tag{4.3}$$

, which implies the maximum of $\lambda$ that has closest distribution with observed histogram is:

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} X_i}{n} = \bar{X} \tag{4.4}$$

### 4.2.3 Local Linear Regression (LLR)

LLR is one of the kernel smoother techniques for estimating a real value function, when no parametric model for this function is known. LLR combines much of the simplicity of linear least square regressions by fitting the line about the given $k$ number of points with the $N$ number of observed points. In the prediction model, the $k$ is corresponded to the number of $TP$s on $PP$s and $N$ is equivalent to the number of history parameters we will employ for the prediction. After fitting the line at every given point, the estimation functions $\hat{Y}(TP_k)$ are achieved as a value function with the $k$ numbers of values. $K_{h\lambda}(X_u, X)$ be a kernel defined by:

$$K_{h\lambda}(X_u, X) = D(\frac{\|X - X_u\|}{h_\lambda(X_u)}) \tag{4.5}$$

The $D()$ is a positive real valued function in (4.5), which is decreasing when the distance between $X$ and $X_u$ increases. The $X_u$ is the given points and the $X$ is one of the observed data around $X_u$. Commonly used kernels include the Epanechnikov, biweigh and Gaussian function. For one dimension data, least-square method is employed for obtaining function value on the $X_u$.

$$\min_{\alpha(X_u),\beta(X_u)} \sum_{i=1}^{N} K_{h\lambda}(X_u, X)\Big(Y(X) - \alpha(X_u) - \beta(X_u)TP_i\Big)^2 \tag{4.6}$$

The $N$ is the number of history parameter near $X_u$ that we will employ in (4.6). Since we obtain parameters in each $TP_i$ by using MLE, the minimum of $\alpha(X_u)$ and $\beta(X_u)$ can be achieved by solving the weighted least square problem (4.6). If we assume the estimation function on $X_u$ is $\hat{Y}(X_u) = \alpha(X_u) + \beta(X_u)X_i$, the closed form solution of the estimation function is like:

$$\hat{Y}(X_u) = (1, X_u)(B^T W(X_u) B)^{-1} B^T W(X_u) y \tag{4.7}$$

where:

$$y = (Y(X_1), \ldots, Y(X_N))^T \tag{4.8}$$

$$W(X_0) = diag\Big(K_{h\lambda}(X_u, X_i)\Big)_{N\times N} \tag{4.9}$$

By repeating this process about all given $k$ points, $X_u$, we can get real value estimation functions $\hat{Y}(X_u)$ about $k$ points.

### 4.2.4 Cyclic Window Learning Algorithm

We describe the algorithm that predicts parameters of the probability distribution of the future target periods by using a cyclic window approach. We assume the dataset has enough past data for the prediction and determined a probability distribution model for MLE. The algorithm will employ LLR to predict the probability distribution parameters of the future target periods and MLE for updating dataset.

**Algorithm 2** Cyclic Window Learning Algorithm

**Require:** $PData_{m\times l}$,$TPdata_t$, $m$, $n$, and $l$
**Ensure:** $PT_t$ and $AT_t$
1: $t = 1$,$p = 1$, and $w = 1$
2: **while** System operate **do**
3: **if** $p < n$ **then**
4: $UT_t = PData_{(m-n-p)\times l} : PData_{m\times l}$, $PData_{1\times l} : PData_{p\times l}, \forall l$
5: **else**
6: $UT_t = PData_{(p-n)\times l} : PData_{p\times l}, \forall l$
7: **end if**
8: Implement LLR in terms of $UT_t$
9: $PT_t = \hat{Y}(UT_p)$ and select $k_{th}$ value
10: Update databased with actual parameter
11: $AT_p = MLE(TPdata_t)$
12: Update databased with $AT_p$
13: $PData(p, w) = AT_p$
14: $t = t + 1$
15: **if** $p < m$ **then**
16: $p = p + 1$
17: **else**
18: $p = 1$
19: **if** $w < l$ **then**
20: $w = w + 1$
21: **else**
22: $w = 1$
23: **end if**
24: **end if**
25: **end while**

The Algorithm 2 obtains the predicted parameters of target period ($PT$) for the prediction and actual parameters of target period ($AT$) to update dataset at every time period. The $m$ means the number of $TP$s included in a $UP$, which is equivalent to a window size. The $n$ is the number of $TP$s during a $PP$. The $l$ represents how many cycles of $PP$s will be stacked on the prediction dataset ($PData$). The $PData$ is the prediction dataset has the $m \times l$ dimension. The $TPdata_t$ means observed requests during an interval of the $TP$ at time $t$. The prediction model will obtain parameters of the $TP$ at time $t$ with MLE by using this data, $TPdata_t$.

We initialize variables: $t$, $p$, and $w$ (line 1). The $t$ represents how many unit periods are passed after the algorithm starts and the $p$ is a corresponding position of the $TP$ about time $t$. Since we return to the initial position on the $PP$ when the $p$ reaches the end of the $PP$, the $p$ becomes a cyclic number from 1 to $m$. The $w$ is a row position on the $m \times l$ $PData$. Too much data requires the complexity of the prediction and consumes too much time for the prediction. Therefore, we stack only the appropriate number of cycles on $PData$ by replacing the old data. The $w$ is which row position in the $PData$ will be updated.

First, the algorithm collects the data for the prediction from the $PData$ to the $UT_t$ (line 2 − line 7). If the position $p$ is less than the window size $n$, we need to employ the data from the end of the $PData$ because the $p$ is cyclic. So we collect the data from $m - n - p_{th}$ to $m_{th}$ columns' data and from the first to $p_{th}$ columns of $PData$ (line 3 − line 4). If the $p$ is greater than the window size $n$, we collect previous $n$ columns data from the point $p$ on the $PData$ (line 5 − line 7).

We implement LLR about the collected data, $UT_t$, and obtain the prediction value about time $t$ (line 8 − line 9). In order to update the $PData$, we need to obtain the actual probability distribution parameters of observed requests. We apply MLE about the data $TPdata_t$ and update corresponding data block in $PData$ (line 10 − line 13). We update the position on the $PData$ for next prediction. We update the $t$ for predicting next time point (line 14). We just increase the $p$ if the position of the $p$ is still on $PP$. If the $p$ exceeds the size of $PP$, $m$, it goes back to the initial position of the $PP$. We also update the $w$ when $p$ goes back to the initial position because that means one row of $PData$ is filled with new data. The $w$ increases when the $p$ increases. However, the $w$ becomes one if the $w$ exceeds $l$, which is vertical size of the

matrix $PData$ (line 15 − line 23).

### 4.2.5 Experiment

#### 4.2.5.1 Google Cluster Data Analysis

The experiment is implemented with Google cluster-usage traces data [38]. Google cluster is a set of computing resources composed of thousands of machines. A job is composed of several tasks which can be processed separately. So each task will be a unit of a process. We consider the number of task arrivals, CPU requests, and Memory requests of tasks. Each task has a timestamp which represents when the task arrives at the cluster. Therefore, the distribution of the number of task arrivals can be observed by using the timestamp. The cluster data also contains the CPU and memory requests of each task. The CPU requests show core counts or core-seconds/second of tasks and the memory requests represent how much bytes each task requires. The cluster starts measurement 600 seconds after the system is operated and has accumulated data for one month approximately. We select a random point to collect data for the prediction model. One weak data is sampled as a training dataset for the prediction modeling and the following week data is employed to test the accuracy of the prediction model.

#### 4.2.5.2 Traffic Pattern Analysis

All experiments are conducted by Matlab. Basically, user requests have a regular pattern. Requests increases during the daytime and weekdays more than the nighttime and weekend. We could observe some patterns by analyzing the Google trace data. The start time of trace is randomly chosen in data. So we do not know what the exact date or time of measurement points. However, our assumption is arriving tasks will show regular patterns in every interval.

Figure 4.4 shows patterns of task arrivals during a week. The number of task arrivals is counted for every one hour, so we could observe how many tasks arrive at the cluster in every hour during the week. The hourly pattern shows high peaks for the first 4 hours and the last 10 hours. So we can assume daytime starts around the tenth hours from the measurement because the number of tasks is increased from the tenth hour and end at the fourth hour in Figure 4.4.

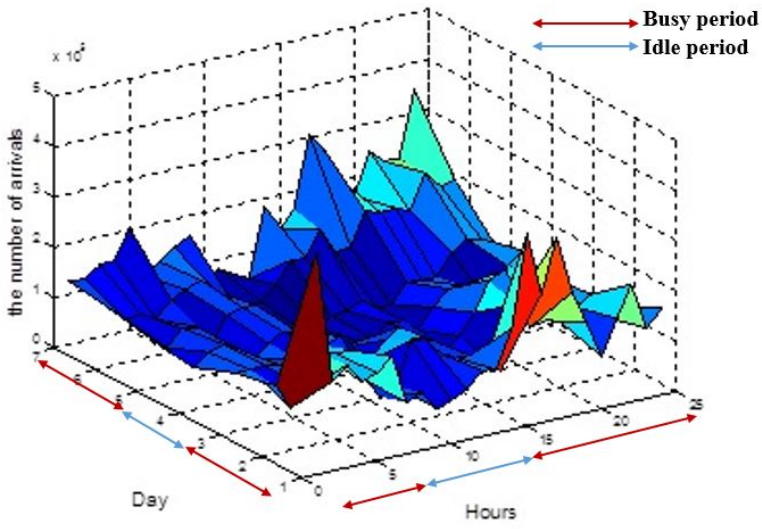


Figure 4.4: Prediction dataset

If we see the daily pattern, the first two days and the last two days show higher requests. In the same way, we can assume weekdays start on the fourth day from the measurement by assuming requests increases during weekdays rather than the weekend.

The daily pattern analysis of task arrivals presents obvious patterns of the incoming requests. Therefore, the cyclic data collection should be an effective approach for the prediction.

The daily pattern analysis of the CPU and memory requests also have demonstrated the similar patterns with task arrivals. The CPU and memory requests have had high requests when the number of task arrivals increases and had low requests during free periods.

According to our observation, we have decided to set the $PP$ to one week because we could observe hourly and daily patterns of requests repeatedly during every week.

#### 4.2.5.3 Histogram Analysis of Data

According to the experiment, we found that too short $TP$ is not enough to observe apparent patterns of distribution of requests. Thus, the $TP$ is set to 30 minutes based on empirical observation. The prediction model predicts distribution parameters in every 30 minutes.

As patterns are observed in Figure 4.4, the number of task arrivals show an apparent pattern depending on the time. Figure 4.5 represents the number of task arrivals in every 30 minute

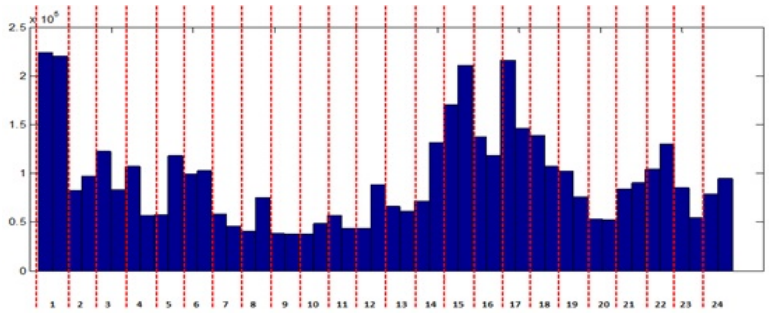

Figure 4.5: The number of tasks during the day

during the first day. For the first 4 hours and the last 10 hours, Figure 3 presents high rate of incoming tasks.

Histograms of each target period are different depending on the total number of task arrivals. Histograms have high peaks in the more right side during the busy hours and they have high peaks in the more left side during the free hours.

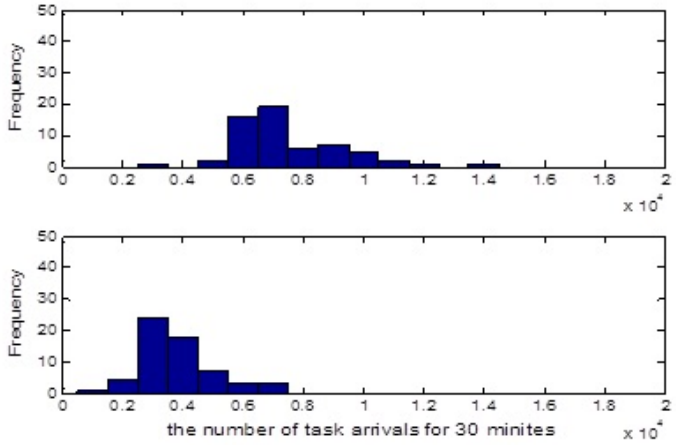


Figure 4.6: Distribution of the first day

Figure 4.6 is a histogram of the first hour. The first histogram exhibits the distribution of task arrivals during the first half hour and the second histogram presents the distribution of the second half hour. The first histogram has a peak in more right side than the second histogram because the cluster received higher task arrivals during the first half hour than the second half hour.

This trend is similar to the Poisson distribution. The Poisson distribution has a high peak

in the more right side when the rate parameter $1/\lambda$ is high. Therefore, the observed data will be fitted to the Poisson distribution by using MLE to obtain parameters of the Poisson distribution in every duration.

#### 4.2.5.4 Maximum Likelihood Estimation of Distribution

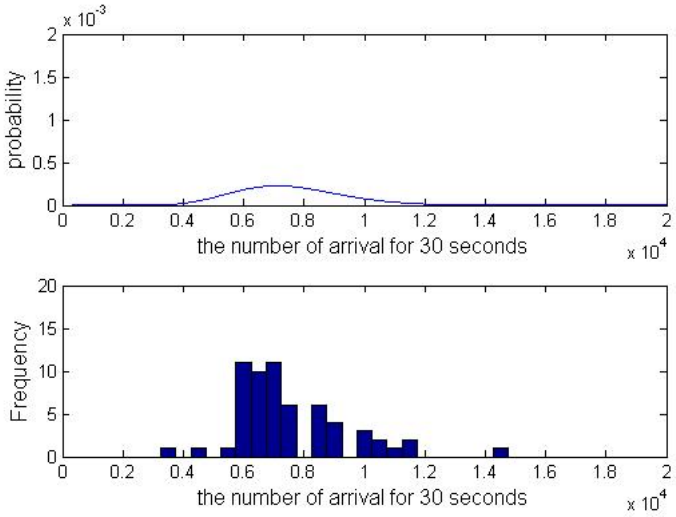


Figure 4.7: MLE of the first hour

MLE is employed to obtain parameters of the Poisson distribution in every target period. The estimated Poisson distribution are achieved about the first half hour histogram by using MLE in Figure 4.7. The first graph shows the Poisson distribution with estimated parameters and the second graph is a histogram of task arrivals during the first half hour. We can observe the estimated Gamma distribution has a similar distribution with the histogram of data. The prediction model implements MLE in every 30 minute and saves them to the prediction dataset.

#### 4.2.5.5 Time Dependent Parameter Prediction

Parameters are induced from the request data in every 30 minute. Parameters generated at the same time point on the $PP$ are stacked in the same column of the prediction dataset. Predicted parameters are achieved by implementing LLR about the corresponding $UP$. For example, if we implement LLR about the $UP$ including the first to the tenth $TP$ to predict the duration corresponding to tenth $TP$, the last point value of LLR function becomes the

prediction value of the tenth target period. Prediction values are changed depending on how to set the utilization periods and how much bandwidth we adopt for LLR. The bandwidth represents how many near data are included when we predict the function value of a certain point.

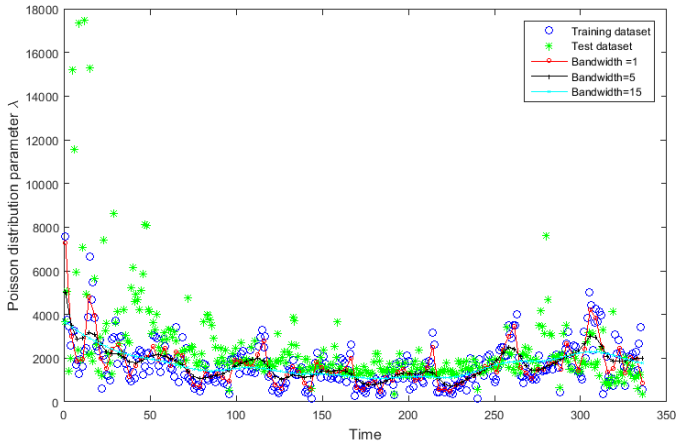


Figure 4.8: Poisson distribution parameter $\lambda$ estimation of arrival tasks

Figure 4.8 represents the prediction of the Poisson distribution parameter $\lambda$ of arrival tasks during a week. Since we set the $TP$ to 30 minutes, we have 336 $TP$s during the week. Blue points represent the parameter of the training dataset in each $TP$ and green points are the parameter of the test dataset in each $TP$. Solid lines represent predicted parameters depending on the bandwidth. We set the $UP$ to 25 hours in Figure 6, which means that prediction value is obtained based on last 25 hours $TP$s parameter data. The Poisson distribution parameter $\lambda$ is the shape parameter of the distribution. We can recognize that the graph has a repeated pattern regularly. It has seven high peaks in the graph, which means similar patterns repeated every day during a week.

#### 4.2.5.6 Error Assessment

In order to quantify an accuracy of the prediction, we measure Mean Absolute Percentage Error (MAPE) between the prediction data and the test dataset. MAPE expresses an error rate as a percentage. So we can compare the prediction accuracy of task arrivals, CPU requests,

and memory requests with a normalized error rate value.

$$MAPE = \frac{1}{n}\sum_{j=1}^{n}\frac{|P_j - T_j|}{T_j} \tag{4.10}$$

The $P_j$ is a predicted value of a target value, $T_j$. MAPE value is equal to zero when the prediction model is the perfect fit to the target value and increased when the prediction is not properly fit to target values.

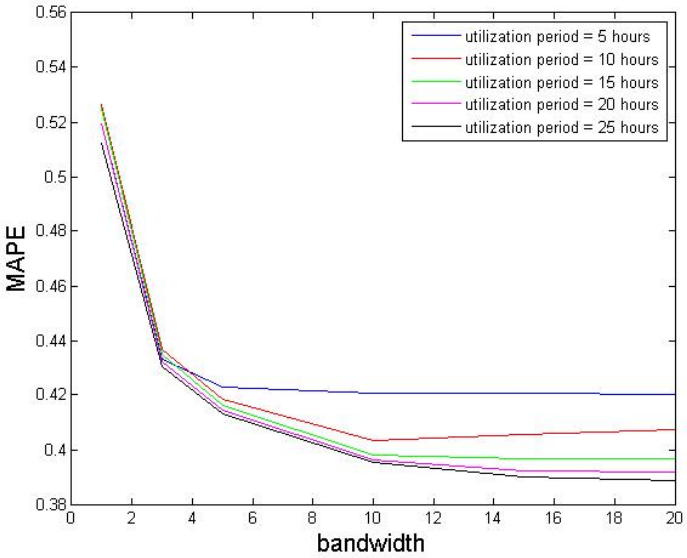


Figure 4.9: MAPE measurement of task arrivals prediction

Figure 4.9 is MAPE measurement graph of the Poisson distribution parameter $\lambda$. The parameter $\lambda$ has MAPE range between 0.3885 and 0.5194. If we consider the ideal state of MAPE is zero, the prediction model has enough prediction accuracy. The prediction model could achieve a higher accuracy with the longer UP, which is equivalent the window size because increasing the UP means employing more previous data for the prediction. However, the large UP requires more complexity of a computation and consumes more time. In other words, a proper selection of the UP is required to satisfy both of the prediction accuracy and the computation time. Choosing the best bandwidth is also an important issue in order to reduce the prediction error. Too small bandwidth causes very spiky estimates while large bandwidth leads over smoothing. If data values are spread widely, the smaller bandwidth will not acquire the higher prediction accuracy.

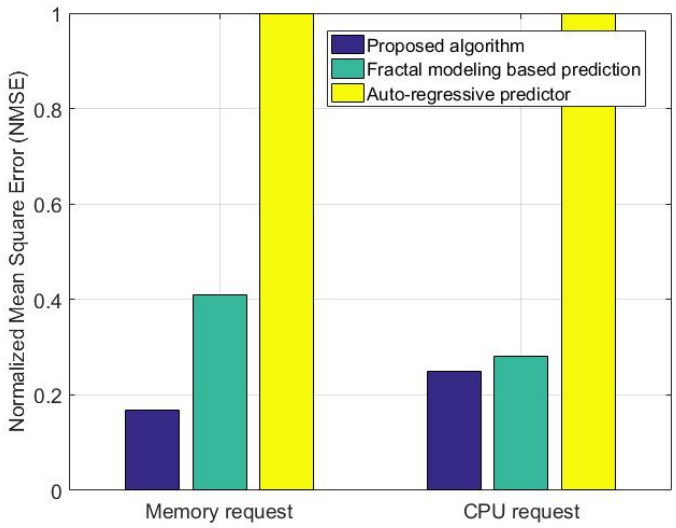


Figure 4.10: NMSE comparison between proposed algorithm and reference models

The proposed algorithm is compared with fractal differential equation modeling based prediction method proposed in [38]. The Mean Square Error (MSE) of the proposed algorithm and comparison prediction model is measured for CPU and memory requests and normalized by baseline prediction algorithm. Auto-regressive predictor is used for baseline model by employing 16 previous time slot value, which is equivalent to using 16 UP in the proposed algorithm. In memory request prediction, the proposed model achieves 25% reduced MSE compare to fractional modeling based predictor and 84% reduced MSE than the auto-regressive predictor. In CPU request prediction, the proposed algorithm shows slightly advanced prediction accuracy than fractal modeling based predictor, 3% reduced MSE but it reduces 75% MSE compare to the baseline prediction model.

## 4.3 Adaptive data center activation

We formulate an adaptive data center activation problem as an MILP. Our objective is to minimize the activation power, port power, and memory power consumption in the data center while guaranteeing QoS. This model decides the active state of switches and hosts by using binary variables and determines load distribution over switches and hosts with continuous

variables. Based on allocated workload in switches and servers, controllers can determine how many memory modules need to be on. We assume that jobs arrive at the data center according to a Poisson distribution with rate $\lambda$. We define the following variables:

- $C_{ij}$: working state of $i_{th}$ group $j_{th}$ core switch; binary variable.
- $A_{ij}$: working state of $i_{th}$ POD $j_{th}$ aggregation switch; binary variable.
- $T_{ij}$: working state of $i_{th}$ POD $j_{th}$ ToR switch; binary variable.
- $H_{ij\_ToR_{mn}}$: working state of $i_{th}$ POD $j_{th}$ host connected to $ToR_{mn}$ ToR switch; binary variable.
- $M_{C_{ij}}$, $M_{A_{ij}}$, $M_{T_{ij}}$, $M_{H_{ij\_ToR_{mn}}}$: The number of operating memory modules in core, aggregate, ToR switches, and hosts; integer variables.
- $\lambda_{C_{ij}}$, $\lambda_{A_{ij}}$, $\lambda_{T_{ij}}$, $\lambda_{H_{ij\_ToR_{mn}}}$: Job arrival rate to core, aggregate, ToR switches and hosts; continuous variable.
- $\lambda_{C_{ij\_l}}$: Job arrival rate to $i_{th}$ POD $j_{th}$ core switch $l_{th}$ port; continuous variable.
- $\lambda_{A_{ij\_l}}$: Job arrival rate to $i_{th}$ POD $j_{th}$ aggregation switch $l_{th}$ port; continuous variable.
- $\lambda_{T_{ij\_l}}$: Job arrival rate to $i_{th}$ POD $j_{th}$ ToR switch $l_{th}$ port; continuous variable.
- $P^{static}_{C_{ij}}$, $P^{static}_{A_{ij}}$, $P^{static}_{T_{ij}}$, $P^{static}_{H_{ij\_ToR_{mn}}}$: Static power consumption of core, aggregate, ToR switches and hosts during a unit time; constant.
- $P^{dynamic}_{C_{ij}}$, $P^{dynamic}_{A_{ij}}$, $P^{dynamic}_{T_{ij}}$, $P^{dynamic}_{H_{ij\_ToR_{mn}}}$: Dynamic power consumption of core, aggregate, ToR switches and hosts on the full utilization state during a unit time; constants.
- $P^{port}_{C_{ij}}$, $P^{port}_{A_{ij}}$, $P^{port}_{T_{ij}}$: Power consumption of each port in core, aggregate, and ToR switche per job; constants.
- $P^{memory}_{C_{ij}}$, $P^{memory}_{A_{ij}}$, $P^{memory}_{T_{ij}}$, $P^{memory}_{H_{ij\_ToR_{mn}}}$: Memory power consumption of core, aggregate, ToR switches and hosts during a unit time; constants.
- $\mu_{C_{ij}}$, $\mu_{A_{ij}}$, $\mu_{T_{ij}}$, $\mu_{H_{ij\_ToR_{mn}}}$: service rate of core switch; constants.
- $\lambda$: Job arrival rate to data center; constant.

### 4.3.1 Objective Function

*Minimize*

$$p^{activation} + p^{port} + p^{memory} \tag{4.11}$$

The objective function includes activation power, port power, and memory power consumption during a unit time. Activation power is the power consumption necessary for operating switches or hosts. Port power is the power consumption for transmitting a job through the

port. We assume that ports are in the idle state when they do not transmit jobs and do not consume power in the idle state. So port power is calculated depending on the number of jobs they transmit. Memory power is determined depending on how many memory modules are powered. If we turn on more memory modules than those required in a switch or a host, this will waste energy. Therefore, we propose to turn on the appropriate number of required memory modules based on statistical estimation.

#### 4.3.1.1 Adaptive Power

$$
\begin{aligned}
p^{activation} = & \sum_{i=1}^{k/2}\sum_{j=1}^{k/2} C_{ij}(P_{C_{ij}}^{static} + P_{C_{ij}}^{dynamic}\frac{\lambda_{C_{ij}}}{\mu_{C_{ij}}}) \\
& + \sum_{i=1}^{k}\sum_{j=1}^{k/2} A_{ij}(P_{A_{ij}}^{static} + P_{A_{ij}}^{dynamic}\frac{\lambda_{A_{ij}}}{\mu_{A_{ij}}}) \\
& + \sum_{i=1}^{k}\sum_{j=1}^{k/2} T_{ij}(P_{T_{ij}}^{static} + P_{T_{ij}}^{dynamic}\frac{\lambda_{T_{ij}}}{\mu_{T_{ij}}}) \\
& + \sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} H_{ij\_ToR_{mn}}(P_{H_{ij\_ToR_{mn}}}^{static} \\
& \qquad + P_{H_{ij\_ToR_{mn}}}^{dynamic}\frac{\lambda_{H_{ij\_ToR_{mn}}}}{\mu_{H_{ij\_ToR_{mn}}}})
\end{aligned}
\tag{4.12}
$$

We can calculate the activation power by considering the static and dynamic power of switches and hosts. The static power is a constant power required for activating switches and hosts. The static power can be calculated by multiplying the binary variables that represent the state of each switch and host by the static power consumption constant of each switch and host. The dynamic power increases proportionally to the utilization rate of switch and host. Thus, the dynamic power can be computed by multiplying the utilization, $\frac{\lambda}{\mu}$, and dynamic power consumption constant of each switch and host. The dynamic power expression includes a non-linear term, corresponding to the multiplication of the binary variable and continuous variable. This non-linear term can be linearized by using a simple transformation introduced.

#### 4.3.1.2 Port Power

$$p^{port} = \sum_{i=1}^{k/2}\sum_{j=1}^{k/2}\sum_{l=1}^{k} \lambda_{C_{ij\_l}} P_{C_{ij}}^{port} + \sum_{i=1}^{k}\sum_{j=1}^{k/2}\sum_{l=1}^{k/2} \lambda_{A_{ij\_l}} P_{A_{ij}}^{port} + \sum_{i=1}^{k}\sum_{j=1}^{k/2}\sum_{l=1}^{k/2} \lambda_{T_{ij\_l}} P_{T_{ij}}^{port} \tag{4.13}$$

Port power should only be considered in switches. The problem decides how each switch will distribute arriving jobs through which port. Therefore, we can measure the port power consumption by multiplying the amount of jobs passing through each port and the power consumption per job of each port.

#### 4.3.1.3 Memory Power

$$\begin{aligned} p^{memory} &= \sum_{i=1}^{k/2}\sum_{j=1}^{k/2} M_{C_{ij}} P_{C_{ij}}^{memory} \\ &+ \sum_{i=1}^{k}\sum_{j=1}^{k/2} M_{A_{ij}} P_{A_{ij}}^{memory} + \sum_{i=1}^{k}\sum_{j=1}^{k/2} M_{T_{ij}} P_{T_{ij}}^{memory} \\ &+ \sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} M_{H_{ij\_T_{mn}}} P_{H_{ij\_ToR_{mn}}}^{memory} \end{aligned} \tag{4.14}$$

We will estimate the number of jobs in each switch and host memory by using a queuing model. Then we can decide how many memory slots we need to operate based on the estimated number of jobs in each switch and host. After we decide the number of working memory slots, we can evaluate memory power consumption by multiplying the number of memory slot and memory power consumption of each memory slot in each switch and host.

### 4.3.2 Load Distribution Constraint

Workload should be distributed rationally in data centers. Since we are proposing to turn off parts of the switches and hosts in the data center, we have to decide exactly which port will transfer jobs to the lower layers. For example, if the core switch transfers jobs to an inactive

aggregation switch, we will not be able to allocate those jobs to hosts. Thus, load distribution constraints decide how we distribute the workload to low layer switches through which port.

#### 4.3.2.1 Load Distribution in Each Layer

The summation of workload in each layer should be equivalent to $\lambda$, the total arriving workload to the data center. The inter-POD traffic does not need to go through the core switch layer in traditional data centers. However, all traffic goes through the core switch layer in SDN data center architecture. Therefore, the summation of traffic in each layer will be the same. The purpose of this constraint is to allocate the workload to working switches and hosts while guaranteeing that we forward all requests to hosts.

$$\sum_{i=1}^{k/2}\sum_{j=1}^{k/2} C_{ij}\lambda_{C_{ij}} = \lambda \tag{4.15}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} A_{ij}\lambda_{A_{ij}} = \lambda \tag{4.16}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} T_{ij}\lambda_{T_{ij}} = \lambda \tag{4.17}$$

$$\sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} H_{ij\_ToR_{mn}}\lambda_{H_{ij\_ToR_{mn}}} = \lambda \tag{4.18}$$

Equations (4.15)-(4.18) are multiplications of binary variables and continuous variables, so they are non-linear equation. Therefore we linearize those non-linear constraints. When $x$ is a binary variable and $y$ is a continuous variable that is greater than or equal to zero, we can linearize their product by replacing it by a continuous variable $t$ with the constraints shown below.

$$0 \leq t \leq \max(y) \cdot x \tag{4.19}$$

$$y - (1 - x) \cdot \max(y) \leq t \leq y \tag{4.20}$$

The maximum of continuous variables, $\lambda_{C_{ij}}$, $\lambda_{A_{ij}}$, $\lambda_{T_{ij}}$, and $\lambda_{H_{ij\_T_{mn}}}$ is $\lambda$ because it is the maximum workload that can be allocated to each switch and host, and is the summation of all

workloads in the same layer. Therefore, we can linearize non-linear constraints (4.15), (4.16), (4.17), and (4.18) with this linearization technique.

#### 4.3.2.2 Link Distribution

As mentioned before, it is important to determine which port to use for transferring jobs. If we use a port connected to an inactive switch or host for transmitting that job, we will lose the request.

$$\lambda_{C_{ij}} = \sum_{l=1}^{k} \lambda_{C_{ij\_l}}, \forall i, j \tag{4.21}$$

$$\lambda_{A_{ij}} = \sum_{l=1}^{k} \lambda_{C_{lj\_i}}, \forall i, j \tag{4.22}$$

$$\lambda_{A_{ij}} = \sum_{l=1}^{k/2} \lambda_{A_{ij\_l}}, \forall i, j \tag{4.23}$$

$$\lambda_{T_{ij}} = \sum_{l=1}^{k/2} \lambda_{A_{il\_j}}, \forall i, j \tag{4.24}$$

$$\lambda_{T_{ij}} = \sum_{l=1}^{k/2} \lambda_{T_{ij\_l}}, \forall i, j \tag{4.25}$$

$$\lambda_{T_{ij\_l}} = \lambda_{H_{il\_ToR_{ij}}}, \forall i, j, ToR_{ij} \tag{4.26}$$

Equations (4.21), (4.23), and (4.25) regulate the distribution from each switch to lower layers. Arriving job requests to each switch should be equivalent to the summation of workloads to go out through their ports.

Equations (4.22), (4.24), and (4.26) decide how arriving jobs are handled by each switch. Summation of the number of jobs going into switches should be equal to the summation of arriving jobs from upper layer through connected port.

Through this constraint, we can decide which port will transfer how many jobs in each link. We can obtain an optimal workload distribution strategy with these constraints.

#### 4.3.2.3 Distribution Restriction

Jobs cannot be distributed to inactive switches and hosts. Constraints (4.15)-(4.18) regulate the summation of jobs in each layer but it can allocate jobs to deactivated switches because

the summation of workload is still the same if the switch is turned off. Therefore, we need constraints that prohibit workloads from being allocated to inactive switches and hosts, and these are as follows:

$$\lambda_{C_{ij}} \leq \lambda \cdot C_{ij}, \forall i, j \tag{4.27}$$

$$\lambda_{A_{ij}} \leq \lambda \cdot A_{ij}, \forall i, j \tag{4.28}$$

$$\lambda_{T_{ij}} \leq \lambda \cdot T_{ij}, \forall i, j \tag{4.29}$$

$$\lambda_{H_{ij\_ToR_{mn}}} \leq \lambda \cdot H_{ij\_ToR_{mn}}, \forall i, j, m, n \tag{4.30}$$

These constraints force the workload allocated to a switch or a host to zero if the binary variable corresponding to the switch or host is zero.

### 4.3.3 Performance Constraints

In order to guarantee the QoS of data centers, latency constraints will be applied to each switch and host. Switches are assumed to be M/M/1 queues and hosts are M/M/c queues, where c is equivalent to the number of cores of the host using the queuing model, we can obtain the distribution of residual time in the queue. Residual time of a job in queue corresponds to how long each job will stay in switches and hosts. Thus, we will use the residual time distribution to obtain the latency constraint.

#### 4.3.3.1 Latency in a Switch

A switch is modeled as M/M/1 queues. Using the Cumulative Density Function (CDF) of the residual time in M/M/1 queue to obtain the residual time equation in M/M/1 queue, we have

$$F_R(t_{residual}) = 1 - e^{-(\mu - \lambda_{switch})t_{residual}} \tag{4.31}$$

Equation (4.31) represents the probability that the residual time in the queue will be less than $t_{residual}$. Therefore, we can obtain the maximum residual time in the queue by letting $F_R(t_{residual}) = \alpha$, where $\alpha$ is close to 1 and $t_{residual}$ satisfies.

$$1 - e^{-(\mu - \lambda_{switch})t_{residual}} = \alpha \tag{4.32}$$

Slightly manipulating (4.32) we have the upper bound on the residual time in the queue as,

$$t_{residual} = -\frac{\ln(1-\alpha)}{(\mu - \lambda_{switch})} \tag{4.33}$$

Since the residual time cannot exceed the latency constraint, Latency, we have the following constraint.

$$-\frac{\ln(1-\alpha)}{(\mu - \lambda_{switch})} \leq Latency \tag{4.34}$$

which can be expressed in the following linear form,

$$-\ln(1-\alpha) \leq Latency \cdot (\mu - \lambda_{switch}) \tag{4.35}$$

By using (4.35), we can obtain the latency constraint for all switches.

#### 4.3.3.2 Latency in a Host

We use a strategy similar to the above to obtain the delay in hosts. A host is modeled as an M/M/c queue, and the CDF of the residual time in an M/M/c queue is [26]:

$$F_R(t_{residual}) = 1 - \frac{c}{c-\rho} \cdot p_c \cdot e^{-(c\mu-\lambda)t_{residual}} \tag{4.36}$$

$p_c$ is the probability that there are $c$ jobs in a queue, $c$ represents the number of cores in our model, which is usually a small number and $\rho$ represents $\frac{\lambda}{\mu}$. For the most cases of practical interest, all cores will be active and we can assume $p_c$ to be equal to 1 for the purpose of simplifying the problem. Therefore,

$$F_R(t_{residual}) = 1 - \frac{c}{c-\rho} \cdot e^{-(c\mu-\lambda)t_{residual}} \tag{4.37}$$

Again, if the residual time satisfies (4.37) with probability $\alpha$, with $\alpha \approx 1$, then

$$1 - \frac{c}{c-\rho} \cdot e^{-(c\mu-\lambda)t_{residual}} = \alpha \tag{4.38}$$

After simple manipulations of (4.38) we obtain

$$t_{residual} = \frac{\ln \frac{(1-\alpha)(1-\rho)}{c}}{-(c\mu - \lambda)} \tag{4.39}$$

Since the residual time in the queue should be less than the latency constraint, we can get constraint below

$$\frac{\ln \frac{(1-\alpha)(1-\rho)}{c}}{-(c\mu-\lambda)} \leq Latency \tag{4.40}$$

Which can also be written as

$$-\ln(1-\alpha) - \ln(1-\rho) + \ln c \leq Latency(c\mu - \lambda) \tag{4.41}$$

Equation (4.41) is non-linear because it includes the non-linear term, $-ln(1-\rho)$. We will linearize (4.41) by using the piecewise approximation technique used in [27]. Since $\rho$ has a limited domain between 0 to 1, then we can linearize $-ln(1-\rho)$ by calculating a linear function which lower bounds $-ln(1-\rho)$ in each of a number of sub domains of $\rho$. We divide $\rho$ to three sub-domains: [0, 0.75], [0.75, 0.95], and [0.95, 1]. Then, we obtain a linear function that fits each sub-domain, $f_1(\rho)$, $f_2(\rho)$,and $f_3(\rho)$, which lower bounds $-ln(1-\rho)$.

Through approximation, we could achieve three linear functions in each sub-domain, $f_1(\rho) = 1.72\rho$, $f_2(\rho) = 7.1679\rho - 4.1698$, $f_3(\rho) = 40.2359\rho - 35.6308$. By replacing $-\ln(1-\rho)$ in (4.41) by $\max_y f_y(\rho)$, $y$ represents a linear function in the $y$ sub-domain, and we obtain the linear constraint below.

$$\begin{aligned} &\max_y f_y(\rho) - \ln(1-\alpha) + \ln c \\ &\leq Latency \cdot (c\mu_{H_{ij\_ToR_{mn}}} - \lambda_{H_ij\_ToR_{mn}}), \forall i,j,m,n,y \end{aligned} \tag{4.42}$$

#### 4.3.3.3 Service Rate

For the stability of the system, the summation of service rates of each layer should be greater than the total arrival rate of jobs. That is,

$$\sum_{i=1}^{k/2}\sum_{j=1}^{k/2} \mu_{C_{ij}} C_{ij} \geq \lambda \tag{4.43}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} \mu_{A_{ij}} A_{ij} \geq \lambda \tag{4.44}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k/2} \mu_{T_{ij}} T_{ij} \geq \lambda \tag{4.45}$$

$$\sum_{i=1,m=1}^{k}\sum_{j=1}^{k/2}\sum_{n=1}^{k/2} \mu_{H_{ij\_ToR_{mn}}} H_{ij\_ToR_{mn}} \geq \lambda \tag{4.46}$$

### 4.3.4 Memory Constraints

Main memory consumes almost 25% of total energy consumption in the data center. Therefore, minimizing the number of working memory slots will contribute to saving energy consumption in data centers. We decide the number of working memory slots based on the probability that an arriving job is rejected. We assume the job is rejected only when there is not enough memory in the queue.

#### 4.3.4.1 Memory in a Switch

Switches are modeled as M/M/1 queue. If we assume the $n_q+1$ job is rejected in the queue, the probability of losing job is equal to probability of $n_q$ jobs in the queue. So we can calculate the probability of loss like below:

$$p(loss) = p_{n_q} \tag{4.47}$$

We set the probability that an arriving job is successfully queued to $\alpha \approx 1$.

$$p(queuing) = \alpha \tag{4.48}$$

Then we can get the equation below:

$$p(loss) = 1 - p(queuing) = 1 - \alpha \tag{4.49}$$

$p(loss)$ is when there are $n_q$ jobs in queue. So we can calculate the probability of loss like below:

$$p(loss) = 1 - \sum_{i=0}^{n_q-1} p_i \tag{4.50}$$

If we substitute $p(loss)$ in (4.49) with (4.50),

$$\sum_{i=0}^{n_q-1} p_i = \alpha \tag{4.51}$$

$p_i = \rho^i(1-\rho)$ in the M/M/1 queue space .So from (4.51) we can obtain,

$$1 - \rho^{n_q-1} = \alpha \tag{4.52}$$

If we take the log on both sides, we can achieve:

$$n_{switch} = \frac{\ln(1-\alpha)}{\ln(\rho)} + 1 \tag{4.53}$$

Equation (4.53) represents the number of existing jobs in a switch with very high probability $\alpha$. Based on the number of jobs in the queue, we can obtain the number of memory slots we need. If we assume the expected size of a job is $J$ and memory slot size is $MS$, we can establish the following memory constraint.

$$n_{switch} \cdot J \leq M_{switch} \cdot MS \tag{4.54}$$

Since all switches are modeled as M/M/1 queues, we can apply this constraint to all switches and decide the number of working memory slots depending on their allocated workloads.

#### 4.3.4.2 Memory in a Host

Hosts are considered as M/M/c queues. We can use the same approach used with the switch, but instead using the steady state probability of the M/M/c queue. As mentioned before, we assume hosts are always busy when they are turned on. Since c represents the number of cores in a host, we can consider there will be always more than c jobs in the host. Therefore we can use the following equation to obtain steady state probability:

$$\sum_{i=c}^{\infty} p_i \approx 1 \tag{4.55}$$

For $i \geq c$, the steady state of probability in M/M/c queue is given by:

$$p_i = \frac{c^c \rho^n}{c!} p_0 \tag{4.56}$$

where $\rho = \frac{\lambda_{host}}{c\mu}$.

The initial state probability $p_0$ can be found by substituting $p_i$ in (4.55) by (4.56).

$$\sum_{i=c}^{\infty} \frac{c^c \rho^n}{c!} p \approx 1 \tag{4.57}$$

Which reduced to

$$\frac{c^c \rho^c}{c!(1-\rho)} p_0 \approx 1 \tag{4.58}$$

So the initial probability $p_0$ is given by,

$$p_0 \approx \frac{c!(1-\rho)}{c^c \rho^c} \tag{4.59}$$

By using (4.59), we can use steady state probabilities by substituting $p_0$ in (4.56).

If we insert (4.56) in (4.51), we can obtain the number of jobs in hosts in terms of utilization.

$$n_{host} = \frac{\ln(1-\alpha)}{\ln(\rho)} + c \tag{4.60}$$

By using (4.60), we can achieve the required number of memory slots that can accommodate jobs in hosts with the following constraint.

$$n_{host} \cdot J \leq M_{host} \cdot MS \tag{4.61}$$

However, (4.54) and (4.61) are non-linear. Similar to the above piecewise linear approximation, we will divide them into three sub-domains: [0, $\rho_1^*$], [$\rho_1^*$, $\rho_2^*$], and [$\rho_2^*$, 1]. Then we could get a linear function in each sub-domain. The approximation function is different depending on the service rate of switches and hosts. So, if we define the functions that obtain the number of jobs in each switch and host,$n_{switch}$ and $n_{host}$ as $f_n(\rho)$, we can calculate sub-linear functions as:

$$\begin{aligned} f_{n1}(\rho) &= \frac{f_n(\rho_1^*)}{\rho_1^*}\rho \\ f_{n2}(\rho) &= \frac{f_n(\rho_2^*) - f_n(\rho_1^*)}{\rho_2^* - \rho_1^*}(\rho - \rho_2^*) + f_n(\rho_2^*) \\ f_{n3}(\rho) &= \frac{\mu - f_n(\rho_2^*)}{1 - \rho_2^*}\rho + \frac{f_n(\rho_2^*)}{1 - \rho_2^*} \end{aligned} \tag{4.62}$$

$\mu$ is the service rate of each switch and host. Since we approximate the number of jobs by a linear function, we can linearize the constraints (4.54) and (4.61) like below in each switch and host:

$$\max_y(f_{ny}(\rho_{C_{ij}})) \leq \frac{M_{host} \cdot MS}{J} \tag{4.63}$$

$y$ represents the number of approximation functions and $M$ is the number of memory slot that are powered, which is an integer variable.

### 4.3.5 Connectivity Constraints

The architecture constraint is applied to the problem to protect the Fat-tree architecture of data centers. If we deactivate necessary switches for connecting to working hosts, it will cause

the disconnection between the core layer and hosts. Therefore, we define the connectivity constraints that maintain the architecture of data centers.

$$\frac{\sum_{j=1}^{k/2} T_{ij}}{k} \leq \sum_{j=1}^{k/2} A_{ij}, \forall i \tag{4.64}$$

The ToR switches and aggregation switches in the same POD have a complete bipartite connection. Therefore, ToR switch can be connected to the core switch layer even if there is only one working aggregation switch in the same POD. (4.64) activates at least one aggregation switch if ToR switches are activated in the same POD.

$$\frac{k}{2} \sum_{j=1}^{k/2} H_{ij\_ToR_{mn}} \leq T_{mn}, \forall i \tag{4.65}$$

If a host is turned on and is connected to $T_{mn}$ ToR switch, $T_{mn}$ should be turned on as well. Since $T_{mn}$ is a binary variable, the right hand side should not exceed 1. So, we divide the number of turned on hosts by $2/k$ in order to make the upper bound of summation equal to 1.

## 4.4 Simulated Annealing Algorithm

The optimization problem includes binary variables and integer variables in the objective function and constraints. The binary variables decide the state of switches and hosts and the integer variables decide the number of working memory slots in each switch and host. As we include integer variables in the problem, this optimization becomes NP-hard. When we consider a small size data center, the optimization problem can be solved in a reasonable time. However, data centers usually include a large number of hosts and servers. In such cases, it will take a long time to solve. Therefore, we will use the Simulated Annealing algorithm, which is a popular randomized heuristic algorithm that can find a near optimal solution in a reasonable time.

The algorithm requires input parameters: the service rates of hosts and switches ($S_{H_{ij\_ToR_{mn}}}, S_{T_{ij}}, S_{A_{ij}}, S_{C_{ij}}$) total incoming jobs into the data center ($\lambda$), parameter $\beta$ that has a value between 0 to 1, and the maximum number of iterations, *iteration*. We generate the initial allocation of $\lambda$ into hosts while guaranteeing the latency constraint and capacity (line 1). The latency goes to infinity

**Algorithm 3** Simulated Annealing Algorithm

**Require:** $S_{H_{ij\_ToR_{mn}}}, S_{T_{ij}}, S_{A_{ij}}, S_{C_{ij}}, \lambda, \beta$, *iteration*
**Ensure:** $P_{minimum}$, $\lambda_{H_{ij\_ToR_{mn}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$
1: We generate initial allocation to hosts $\lambda_{H_{ij\_ToR_{mn}}}$ depends on service rate of hosts
2: Decide ToR, aggregation and core switches based on allocation into hosts.
3: $\lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$=Switch($\lambda_{h_{ij\_ToR_{mn}}}$)
4: $P_{minimum}$=$Cal\_power(\lambda_{h_{ij\_ToR_{mn}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}})$
5: **while** t$\leq$ iteration **do**
6: Search neighbor allocation $\lambda^{'}_{h_{ij\_ToR_{mn}}}$
7: $\lambda^{'}_{T_{ij}}, \lambda^{'}_{A_{ij}}, \lambda^{'}_{C_{ij}}$=Switch($\lambda^{'}_{H_{ij\_ToR_{mn}}}$)
8: $P_{candidate} = Cal\_power(\lambda^{'}_{H_{ij\_ToR_{mn}}}, \lambda^{'}_{T_{ij}}, \lambda^{'}_{A_{ij}}, \lambda^{'}_{C_{ij}})$
9: r$\leftarrow$rand()
10: **if** $P_{candidate} < P_{minimum}$ or $e^{\frac{p_{minimum}-p_{candidate}}{\beta}} > r$ **then**
11: $\lambda_{H_{ij\_ToR_{mn}}} \leftarrow \lambda^{'}_{H_{ij\_ToR_{mn}}}$
12: $\lambda_{T_{ij}} \leftarrow \lambda^{'}_{T_{ij}}$
13: $\lambda_{A_{ij}} \leftarrow \lambda^{'}_{A_{ij}}$
14: $\lambda_{C_{ij}} \leftarrow \lambda^{'}_{C_{ij}}$
15: $P_{minimum} \leftarrow P_{candidate}$
16: $t$++, $\beta \leftarrow \beta \cdot \alpha$
17: **else**
18: $t$++, $\beta \leftarrow \beta \cdot \alpha$
19: **end if**
20: **end while**
21: OUTPUT: $P_{minimum}$, $\lambda_{H_{ij\_ToR_{mn}}}, \lambda_{T_{ij}}, \lambda_{A_{ij}}, \lambda_{C_{ij}}$

when the allocation to hosts is very close to service rates. Therefore, we decide the allocation rate that does not violate the latency constraint and let the allocation of jobs not exceed that ratio. After we decide the initial allocation to hosts, we can determine which switches will be needed to operate the data center properly. So, the Switch() function finds which switches are required and how many jobs can be allocated to switches based on the assignment result of hosts (line 3). After we decide the allocation of jobs over the entire data center, we calculate the power consumption of the data center based on the result of allocations (line 4). Then we set that solution and the power as a candidate solution. We start finding neighbor solutions in lines 5 to 20. We find the neighbor solution of job allocation into hosts (line 6). In the same method, we can find which switches are required to be activated to reach working hosts. So, we turn on the required switches by using Switch() function (line 7). Then, we can calculate the power consumption of neighbor solution in line 8. If the power consumption of a neighbor solution is less than the power consumption of candidate solution, we replace the candidate solution with the neighbor solution. We also replace candidate solutions with neighbor solutions with some probability to avoid isolation of solution. A random number $r$ is generated between 0 and 1.

If $e^{p_{minimum}-p_{candidate}/\beta}$ is greater than random number $r$, we replace candidate solution with the neighbor solution even if power consumption of neighbor solution is greater than power consumption of candidate solution (line 10 - line 15). In every iteration, $\beta$ is multiplied by $\alpha$, which has a value less than 1, to decrease the $\beta$ in every iteration (line 16 and line 18). The algorithm repeatedly finds neighbor solutions for $t$ iterations. Then, the candidate solution will become a near optimal solution of the problem.

## 4.5 Numerical Results and Discussion

### 4.5.1 Adaptive Data Center Activation

The optimization problem is solved using CPLEX for small data centers and also using the Simulated Annealing algorithm implemented in C. We consider heterogeneous environment data centers having different service rates of hosts. However, all switches have the same performance in each layer. We set the parameters of the model as shown in Table 4.1.

Table 4.1: Value of key parameters

| Variable | Value |
|---|---|
| $P^{static}_{C_{ij}}$, $P^{static}_{A_{ij}}$, $P^{static}_{T_{ij}}$ | 100W, 50W, 50W |
| $P^{dynamic}_{C_{ij}}$, $P^{dynamic}_{A_{ij}}$, $P^{dynamic}_{T_{ij}}$ | 300W, 150W, 150W |
| $P^{static}_{H_{ij\_ToR_{mn}}}$ | [30, 50, 70, 90]W |
| $P^{dynamic}_{H_{ij\_ToR_{mn}}}$ | [100, 150, 200, 250]W |
| $P^{port}_{C_{ij}}$, $P^{port}_{A_{ij}}$, $P^{port}_{T_{ij}}$ | 0.0005W/job |
| $P^{memory}_{C_{ij}}$, $P^{memory}_{A_{ij}}$, $P^{memory}_{T_{ij}}$, $P^{memory}_{H_{ij\_ToR_{mn}}}$ | 25W |

Power consumption parameters of switches are estimated with practical power consumption of data center switch, HP Altoline 6712 Switch Series [51]. Since core switches implement path calculation and management as control plane, they consume more energy than aggregation and ToR switches in the data plane. Switches in the data plane are set to consume half energy consumed by core switches. Hosts have different activation power parameters proportional to their service rate. Our model includes four types of hosts, and therefore hosts have four

levels of activation power consumption depending on their service rates with high performance hosts consuming more power. Identical hosts are allocated to the same POD. Thus, hosts heterogeneity will be across PODs.

The size of requests is randomly generated from a normal distribution with $\mu$=20MB. All switches have two 512MB memory slots and hosts have two 1GB memory slots. The latency constraint of each switch is set to $1\mu s$ based on the reference switch data sheet and the latency constraint of hosts is set to $1ms$ to satisfy the QoS of data center.

The service rate of a core switches are set to 1000 tasks per unit time, and aggregation and ToR switch have the ability to handle 5000 tasks per a unit time. Since we predict the traffic distribution every 30 minutes and decide system activation states, the unit time is 30 minutes in our simulation model. Hosts have 80, 100, 120, and 140 service rates which are corresponding processing speed between 1.6 GHz, lower performance host, to 2.4 GHz, high performance host. Data plane switches, aggregation and ToR switches, have higher service rate than control plane switch,core switches, because they just work for forwarding tasks.

Since the problem is NP-hard, we could not obtain the optimal solution for large size data centers. Optimal solutions only when k is equal to 4 were obtained within a reasonable time using CPLEX. However, the computation time of the problem increased significantly when we increase the size of the data center to k that is greater than 8.

Therefore, we employ Simulated Annealing to obtain a near optimal solution of the problem. The algorithm uses 1000 iterations, $\beta$ is set to 1, and $\alpha$ is 0.9 in Algorithm 1.

Fig. 4.11 compares the solution of the optimal solution and heuristic solution when $k$ is equal to 8. Since the computation time of the problem is unrealistic, we compare the upper and lower bounds of the optimal solution when the gap is around 6%. It has taken around 1300 seconds until we reached the 6% gap and could not get to less than 6% gap even after 3 hours of computation. Data center utilization rate exhibits the incoming rate of tasks compared to maximum service rate capacity of data centers. For stability of data centers, the incoming rate cannot exceed the maximum service capacity of the data center. The Simulated Annealing solution has around 10% difference with the upper bound and 15% difference with the lower bound.

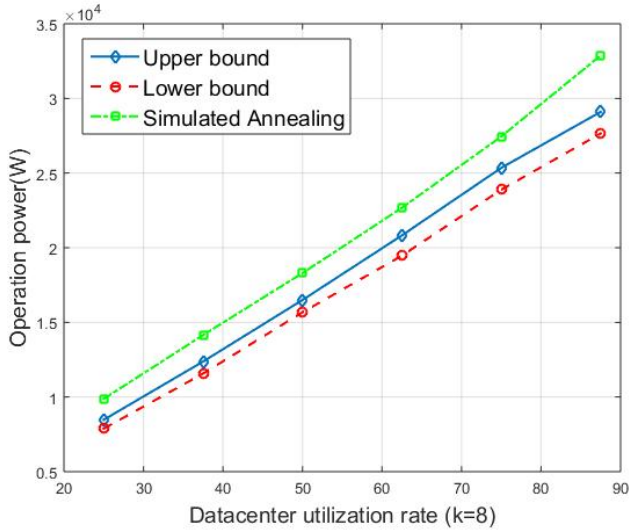


Figure 4.11: Optimal and heuristic solution comparison

Fig. 4.12 shows how much energy we can save with our model. The energy saving rate is calculated by comparing the energy consumption of the heuristic solution and full operation power consumption of data center when the data center operates every switch and host. We measure how much energy we can save depending on the utilization rate and the size of the data center. As utilization rate increases, the energy saving rate is decreased because we cannot turn off many switches and hosts to guarantee the performance of the data center. Also, we can observe that the energy saving rate is increased when the size of data center increases.

The predicted Poisson distribution parameters which are obtained in Fig. 4.8 are employed as the input to the data center in Fig. 4.13. The predicted parameters include 336 prediction points during a week. So each point corresponds to Poisson distribution parameter for 30 minutes. We have made a modification in the data center parameter in order to make it close to Google cluster environment. The port number $k$ is set to 24, and therefore the data center has 24 core switches, 48 ToR and Aggregation switches, and 3456 hosts. Energy consumption and latency parameters are the same as in Table 1 and service rates of switches and hosts are assumed as the same condition.

The first graph in Fig. 4.13 presents predicted parameters during a week and the second graph shows how energy saving rate is changed depends on $\lambda$ in every 30 minutes. In Fig. 4.13,

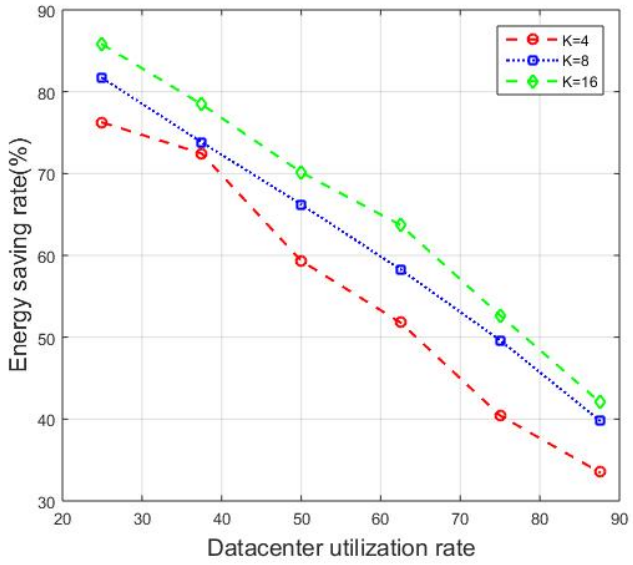


Figure 4.12: Energy saving rate with Simulated Annealing algorithm

we can observe the data center energy saving rate is decreased when the number of arriving tasks increases and it increases when the data center receives fewer requests. The algorithm saves average 47.7 to 48.2% operation energy compared to the full operation state depending on the bandwidth.

Fig. 4.14 shows the difference between actual parameters and predicted parameters. The prediction model makes accurate prediction generally. However, we can observe the prediction model has a weakness in predicting traffic bursts. If the number of requests soars in a short time, the prediction model could not follow that variation. Energy saving rate of the data center model also presents the similar patterns. However, the data center activation model activates switches and hosts by forecasting the high probability traffic of given probability distribution. For example, the $\alpha$ is set to 0.95 in the experiment. Therefore, the data center could cover in some degree of traffic burst. The energy saving rate of proposed dynamic data center activation mode algorithm is compared with Lazy Capacity Provisioning (LCP) algorithm proposed in [55]. The LCP algorithm decides the optimal number of activated servers by considering operating cost and transition cost, the cost of server on-off transition. Since the LCP algorithm has been developed for homogeneous server condition and does not

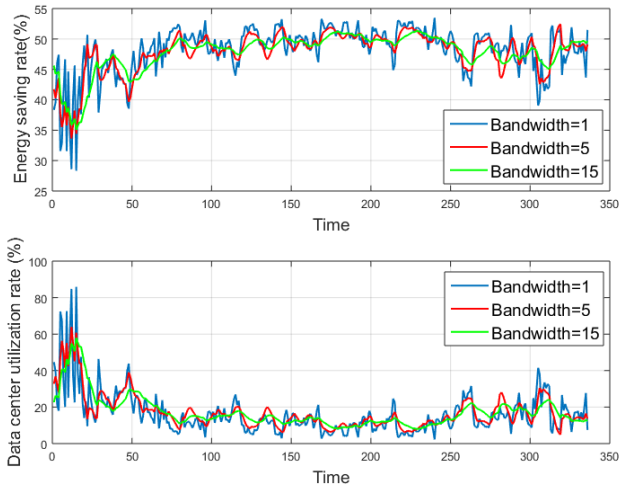


Figure 4.13: Energy saving rate with Simulated Annealing algorithm for different bandwidth values

affect to activation of switch layer, we modified our simulation model to a homogeneous data center. Heterogeneous server service rates are replaced by a unique service rate and power consumption of servers are set to a single value but the total capacity of the homogeneous data center is set to same as the heterogeneous data center model. Since the LCP algorithm does not control the activation of switch layer, all switches are activated if at least one server is activated in the same POD but switches are deactivated if none of the servers are activated in the same POD. In Fig. 4.15, the energy saving rate of the proposed algorithm and the LCP algorithm are compared. The proposed algorithm saves an average of 9.1% more energy compared to the LCP algorithm. Also, we can observe that the LCP algorithm does not perform well for traffic burst. During the time slot between 5 to 25, we can observe that the data center utilization rates vary a lot. Since the LCP algorithm minimizes operation cost and transition cost, the data center tries to keep the current state of activated servers. Since the traffic burst is a common situation in cloud computing environments, considering transition cost is not the appropriate model for energy saving purposes.

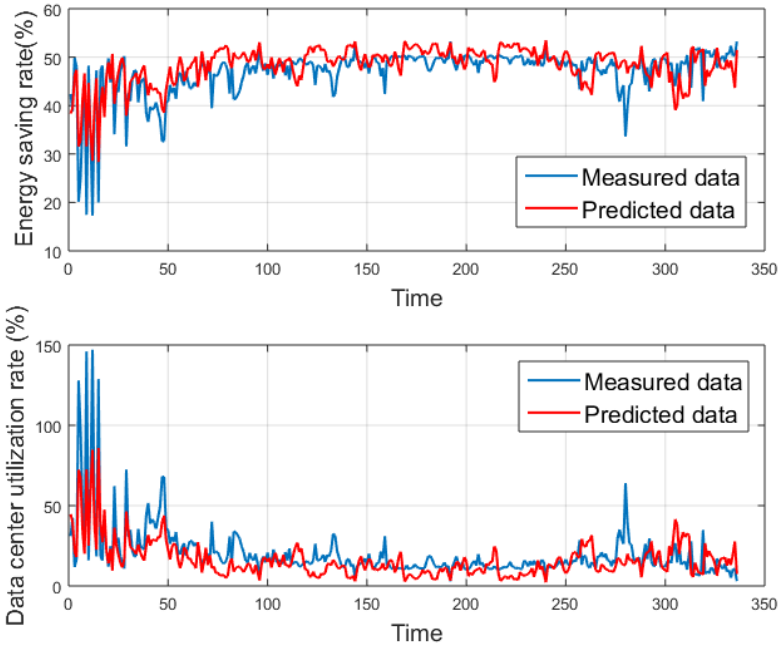


Figure 4.14: Energy saving rate comparison between predicted and measured user requests

## 4.6 Chapter Summary

This Chapter proposed adaptive data center activation model with request prediction algorithm. With accurate prediction model, the data center could save the energy in the more efficient way. The prediction model exhibits accurate prediction compared to another prediction algorithms and the adaptive activation model presents flexible energy saving rate depending on incoming rates of requests. When the model is tested with predicted data based on real data, we could save 30 to 50 percent of energy compared to full operation environment.

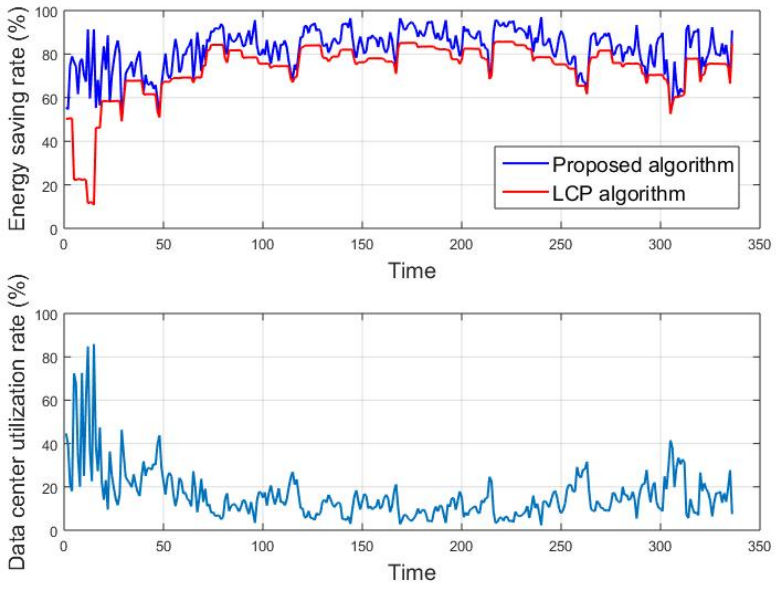


Figure 4.15: Energy saving rate comparison with LCP algorithm

# CHAPTER5. NETWORK FUNCTION VIRTUALIZATION DEPLOYMENT IN CLOUDS

The European Telecom Standards Institute (ETSI) proposed the concept of Network Function Virtualization (NFV) with the aim of efficient network architecture and network system operation. In the traditional network, network functions are implemented in dedicated physical machines which are deigned for single functionalities. Network services have been provided by connecting these physical machines, so the network architecture has been highly rigid and hard to change. NFV environment provides a more flexible and scalable network configuration and implementation through the softwarization of physical network functions. Network functions are transformed to Virtual Machines (VMs) so that Virtualized Network Functions (VNFs) can be implemented in commodity servers built for common uses.

NFV service is operated through service chaining. NFV Service Chain (NFV-SC) dedicates a set of VNFs having a sequence of processing required for service provider's policy or user requirement. The service provider forces the sequence of VNFs' processing depending on the operation policy.

In this chapter, we develop an efficient NFV MANO algorithm. In Section 5.1, we propose an efficient Virtual Infrastructure Management (VIM) scheme in a single domain infrastructure. In Section 5.2, NFV Orchestrator algorithm is proposed by considering hyper-scale distributed cloud computing infrastructure.

## 5.1 Virtual Infrastructure Management (VIM) Algorithm: NFV Resource Allocation using Mixed Queuing Network Model

We employ BCMP mixed queuing network model to analyze the system performance. Arrival rates to VNFs will be assumed as multi-classes Poisson arrival. Each VNF is modeled as a queue. The connectivity of VNFs are considered as mixed queuing network model including multiple open chains and closed chains. Our purpose is minimizing the expected waiting time of service chains. VNFs' routing probabilities to other network functions are assumed to be known based on the history data and will be learned periodically. In order to minimize the maximum expected waiting time of service chains, we formulate the problem as a convex problem with capacity constraints. The optimal service rates will be assigned to each VNFs to minimize the maximum expected waiting time of service chains while guaranteeing the capacity constraints.

In order to calculate the expected waiting time of mixed queuing network, Mean value Analysis (MVA) technique will be employed. However, it is impossible to solve the optimization problem because the closed queuing network calculation. The mean value computation of closed network, expected queuing length, expected waiting time, and expected throughput, requires recursive algorithm from the empty state of network. Also, the service rate of the stations are required to be given when we start the computation. The service rate is unknown decision variable and our purpose is minimizing the expected waiting time of service chain through optimal service rate allocation. So it is not possible to solve the optimization problem in traditional approach. So we propose the algorithm that allocates efficient service rates to VNFs by using approximation approach.

### 5.1.1 System Model

We employ OpenNF controller model proposed in [66]. OpenNF is composed of two layers: flow manger and NFV state manager as we can see in Figure 5.1. The NFV state manager continuously monitors the state of VNFs and report their states to the flow manager. Then the flow manager sends VNFs status information to SDN switch so that the SDN switches determine flow controls to VNFs. VNFs are allocated to the single server and a hypervisor

allocates resources to VNFs depending on their routing and processing status. Since NFV state manager monitors the status of VNFs continuously, the NFV state manger and VNFs generate closed network per service chain. The NFV state manage generates a signal that keep going around VNFs in the same service chain and report VNFs status to NFV state manager. The number of jobs in closed networks are determined depending on how many VNFs are related to service chain and how many classes are related to each VNF. Multiple closed networks can exist on the server depending on service provider's needs and policy.

After receiving monitoring result from the NFV state manager, flow manager sends monitoring results to SDN switches. Then, SDN switches determine the flow control to VNFs. All incoming and departing flows of VNFs are influenced by SDN switches. SDN switches control the arriving flows to VNFs depending on their status and forward flows to all VNFs included in the related service chain of flows. Then, all flows depart the server through the SDN switches after finishing the process in all required VNFs. Therefore, VNFs form open network with SDN switches per service chain to receive incoming flows and send out the departing flows.

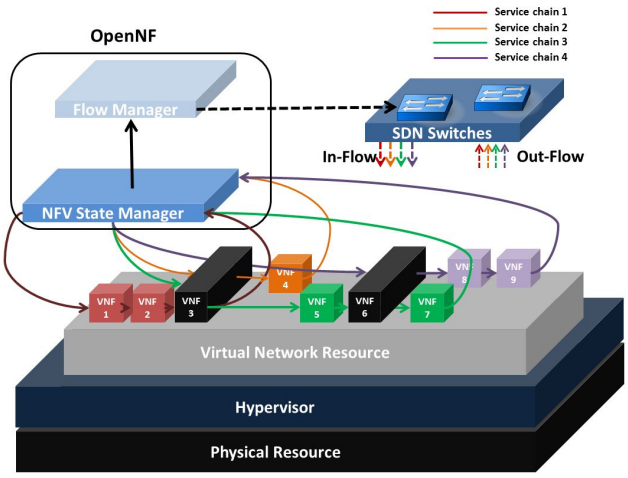


Figure 5.1: System model

The purpose of this Section is to minimize the maximum expected waiting time of service chains in the single server by assigning optimal service rates to VNFs. Each VNF has different routing probability between VNFs, visiting ratio, and incoming rates by open networks. Some VNFs receive high traffic requests from the SDN switch. On the other hand, some VNFs will

not receive frequent requests from users. Thus, it would not be best to assign equal service rates to all virtual network functions. We measure the routing probability or inter-connectivity between network functions continuously and decide the optimal service rates of VNFs.

### 5.1.2 Problem Formulation

Table 5.1 shows the notations used in the following formulations.

Table 5.1: Formulation Notations

| Variable | Value |
|---|---|
| $N_S$ | Population vector of classes |
| $N_C$ | Station population vector of closed classes |
| $N_O$ | Station population vector of open classes |
| $s_{ic}$ | Average amount of loads brought by class $c$ jobs to station $i$ (per visit). |
| $\mu_i$ | Service rate of VNF $i$. Average amount of work completed per unit time. |
| $C_i$ | The inverse of service rate of station i ($1/\mu_i$). |
| $E_i(n)$ | Auxiliary function for the calculation of the effective capacity of mixed station i. $E_i(n) = \frac{1}{(1-L_o C_i)^{n+1}}$ |
| $L_{Oi}$ | Load brought to station $i$ by open class jobs. |
| $\overline{X_{ic}}(N)$ | Throughput of class c jobs at station $i$. |
| $P_{Ci}(N_C, N_S)$ | Marginal length distribution of closed classes. |
| $P_{Oi}(N_O, N_S)$ | Marginal length distribution of open classes. |
| $\overline{n_{ic}}$ | Expected queuing length of closed class $c$ in station $i$. |
| $\overline{n_{iC}}$ | Expected queuing length of closed classes in station $i$. |
| $\overline{n_{io}}$ | Expected queuing length of open class $o$ in station $i$. |
| $\overline{w_{ic}}$ | Expected waiting time of closed class $c$ in station $i$. |
| $\overline{w_{io}}$ | Expected waiting time of open class $o$ in station $i$. |
| $\underline{P}$ | NFV policy matrix. |

#### 5.1.2.1 BCMP Mixed Queuing Network Model

We consider a BCMP mixed queuing network model composed of multi-class jobs and multiple stations, VNFs.

The BCMP queuing network contains a finite number, V, of stations having different disciplines: the First-Come First-Served (FCFS), Processor Sharing (PS), Infinite Servers (IS), and Last-Come First-Served (LCFS). FCFS type stations have a negative exponential service time distribution and others can have a general service time distribution. The only restriction of other disciplines is that the service time distribution should have a rational Laplace transform. The service rate of each station is expressed as the average amount of work completed per time unit.

Open class jobs are incoming traffic to the server that runs VNFs. The classes are classified based on types of the service chain. Each service chain requires different levels of Service Level Agreement (SLA). The delay sensitive applications like video streaming or voice call require higher levels of SLA. On the other hand, delay tolerant applications have relatively lower levels of SLA. Thus, each class of open network has a different average amount of work brought by a single job and it will be considered when we analyze the open networks. Closed networks correspond to monitoring states of VNFs as in Fig. 5.1. Closed networks are as many as the number of service chains.

We employ Mean Value Analysis (MVA) technique to analyze the characteristics of the queuing networks. MVA is an efficient method to analyze the product form expressed queuing network model. MVA employs the mean value equation augmented by Little's law. MVA has a significant merit in terms of computational cost compared to other product form queuing network model analysis methods such as joint distribution analysis or convolution algorithm.

Let C denote the set of closed classes, O denote the set of open classes, and M denote the number of service chains, $O = \{1, 2, \ldots, M\}, C = \{M + 1, M + 2, \ldots, 2M\}$. Since classes are generated as many as service chains for open and closed classes, $O$ and $C$ have the same number of elements.

Let $N_S = [N_1, N_2, \ldots, N_M, N_{M+1}, \ldots, N_{2M}]$ be the population vector of each class in the system. The state $N_M$ is a vector including the possible distribution of jobs over the VNFs, $N_M = [N_{M1}, N_{M2}, \ldots, N_{MV}]$. $V$ stands for the number of VNFs in the server. The number of jobs in closed queue is fixed at a certain time point and the number of jobs in an open class denotes the upper bound for the population of jobs.

The marginal length distribution of closed network is described as (5.1) in [64].

$$P_{Ci}(N_C, N_S) = \sum_{c \subseteq C} s_{ic} C_i \frac{E_i(n_c)}{E_i(n_c - 1)} \overline{X_{ic}}(N_s) P_{Ci}(n_c - 1, N_s - 1) \tag{5.1}$$

The marginal distribution of closed queue is a recursive equation that is defined in terms of $P_{Ci}(n_c - 1, N_S - 1)$. Therefore, the marginal distribution of closed network can be calculated by using a recursive algorithm starting from the empty space of the closed network.

We can obtain the expected number of jobs of closed class by using the following equation.

$$\begin{aligned} \overline{n_{ic}}(N_S) &= \sum_{n_c=1}^{N_C} n_c P_{Ci}(n_c, N_S) \\ &= s_{ic} \overline{X_{ic}}(N_S) \sum_{n_c=1}^{N_C} n_c C_i \frac{E_i(n_c)}{E_i(n_c - 1)} P_{Ci}(n_c - 1, N_s - 1), \forall c \subseteq C \end{aligned} \tag{5.2}$$

We substitute $C_i \frac{E_i(N_C)}{E_i(N_C - 1)}$ by $EC_i$ which defined as $EC_i = \frac{C_i}{1 - L_{Oi} C_i}$ in [64]. By using Little's law, the expected waiting time in a closed network can be calculated by:

$$\begin{aligned} \overline{w_{ic}}(N_S) &= \frac{\overline{n_{ic}}(N_S)}{\overline{X_{ic}}(N_S)} \\ &= s_{ic} EC_i \sum_{n_c=1}^{N_C} n_c P_{Ci}(n_c - 1, N_S - 1), \forall c \subseteq C \end{aligned} \tag{5.3}$$

We rewrite (5.3) as,

$$\begin{aligned} \overline{w_{ic}}(N_S) &= s_{ic} EC_i \sum_{n_c=1}^{N_C} (1 + n_c - 1) P_{Ci}(n_c - 1, N_S - 1) \\ &= s_{ic} EC_i [\sum_{n_c=1}^{N_c} P_{Ci}(n_c - 1, N_S - 1) + \sum_{n_c=1}^{N_c} (n_c - 1) P_{Ci}(n_c - 1, N_S - 1)] \\ &= s_{ic} EC_i (1 + \overline{n_{ic}}(N_S - 1_c)), \forall c \subseteq C \end{aligned} \tag{5.4}$$

$N_S - 1_c$ represents the state that one closed class job is reduced from the state $N_S$. (5.4) shows that the expected waiting time of closed class c is affected by the state $N_S - 1_c$.

The marginal queuing length distribution of open network is described in [64].

$$P_{Oi}(N_O, N_S) = N_O! \Pi_{o \subseteq O} \frac{(\lambda_{io} s_{io})^{N_O}}{N_O} \sum_{n_c=0}^{N_C} \binom{n_c + n_o}{n_c} \Pi_{j=n_c+1}^{n_c+n_o} C_i [E_i(n_c)]^{-1} P_{Ci}(n_c, N_S) \tag{5.5}$$

(5.5) includes the marginal distribution of closed network, which means that the marginal distribution of open network is affected by the marginal distribution of closed network. Therefore the distribution of open network can be calculated after we obtain the marginal distribution of closed network in the state of $N_C$. The expected number of jobs in open network can be described by using (5.5) as described in [64].

$$\begin{aligned}\overline{n_{io}}(N_S) &= \sum_{n_o=1}^{\infty} n_o P_{Oi}(n_o, N_S) \\ &= \lambda_{io} s_{io} \sum_{n_c=0}^{N_C} (n_c + 1) EC_i P_{Ci}(n_c, N_S), \forall o \subseteq O\end{aligned} \tag{5.6}$$

From Little's law, the expected waiting time of open network can be calculated like below:

$$\begin{aligned}\overline{w_{io}}(N_S) &= s_{io} EC_i \sum_{n_c=0}^{N_C} (n_c + 1) P_{Ci}(n_c, N_S) \\ &= s_{io} EC_i (1 + \overline{n_{iC}}(N_S)), \forall o \subseteq O\end{aligned} \tag{5.7}$$

So the expected waiting time of open network can be described based on the expected length of closed network at state $N_S$.

#### 5.1.2.2 Optimal Service Rate Allocation

The purpose of this Section is minimizing the maximum expected waiting time of service chains in the limited resource. Each service chain includes different types and number of VNFs. Therefore, minimizing the maximum expected time of service chains will guarantee the SLA of NFV.

VNFs receive different numbers of requests depending on service chain types and routing probability. In order to minimize the expected waiting time of a service chain, more resources will be allocated to VNFs with relatively heavy traffic. We consider resource allocation in a single server. The resource allocation can be easily managed by the hypervisor depending on VNF requirements.

We assume a policy matrix $\underline{P}$, which is a $M \times V$ matrix that shows policies of each service chain. Each row represents the policy of service chains and each column corresponding to VNFs. If $P_{ij} = 1$, $i_{th}$ service chain needs to be processed in $j_{th}$ VNF. The convex optimization

problem to minimize the maximum expected waiting time service chains can be formulated like below. Since $\underline{P}$ is $M \times V$ matrix and $\vec{w}$ is $V \times 1$ vector, $\underline{P}\vec{w}$ gives $M \times 1$ vectors which is equal to the expected waiting time of each service chain. Thus, the maximum of the vector will give the maximum expected time of service chains.

$Minimize_{\mu_i}$

$$max_M(\underline{P}\vec{w})_{1\times M}$$

*Subject to*

$$C1 : \sum_{i \subseteq V} \mu_i \leq \overline{C}$$

$$C2 : \mu_i \geq \sum_{o \subseteq O} s_{io}\lambda_{io} + \sum_{c \subseteq C} s_{ic}\overline{n_{ic}}(N_S), \forall i$$

$\overline{C}$ denotes the total capacity of the server. The total service rates allocated to VNFs cannot exceed the total capacity of the server in C1. C2 represents the minimum service rate of VNF $i$. For the stability of the system, the minimum service rates of each VNF should be greater than open and closed classes job requests. $\vec{w}$ is the vector that denotes the expected waiting time of each VNF including open and closed networks, $\vec{w} = (w_1, w_2, \ldots, w_V)$ .

$$\overrightarrow{w_i} = \sum_{j \subseteq O \cup C} \overline{w_{ij}}(N_S) \tag{5.8}$$

$\overline{w_{ic}}(N_S)$ and $\overline{w_{io}}(N_S)$ are defined in (5.4) and (5.7). $\overline{w_{ic}}(N_S)$ includes $\overline{n_{ic}}(N_S - 1_c)$, the expected number of jobs in closed network at $N_S - 1_c$ state, and $\overline{w_{io}}(N_S)$ includes $\overline{n_{ic}}(N_S)$ the expected number of jobs in closed queue at $N_S$ state.

However, calculating the expected number of jobs in closed queue requires recursive computation starting from the empty state of closed queue as mentioned in the previous section. The expected queuing length calculation in closed network requires the knowledge of service rate of VNFs. In other words, the minimization of waiting time through service rate allocation cannot be accomplished because we cannot calculate the expected waiting time of closed network without knowledge of service rates, decision variables, of service stations, VNFs. So we employ the Schweitzer Core algorithm described in [65]. The Schweitzer Core algorithm estimates the expected queuing length of closed network by using a iterative algorithm.

The convexity of the problem can be simply proved. The objective function is a linear combination of $\overline{w_{ic}}(N_S)$ and $\overline{w_{io}}(N_S)$ for arbitrary $c$ and $i$ depending on service policy. If we can prove that the individual terms $\overline{w_{ic}}(N_S)$ and $\overline{w_{io}}(N_S)$ are convex, the objective function is a convex function. $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$ will be estimated as a constant by the Schweitzer Core algorithm. Equation (5.4) can be expressed as:

$$\begin{aligned}\overline{w_{ic}}(N_S) =& s_{ic}\frac{C_i}{1 - L_{O_i}C_i}(1 + \overline{n_{ic}}(N_S - 1_c)) \\ =& \frac{s_{ic}}{\mu_i - L_{O_i}}(1 + \overline{n_{ic}}(N_S - 1_c))\end{aligned} \tag{5.9}$$

If we find the second derivative of $\overline{w_{ic}}(N_S)$, $\frac{\partial^2 \overline{w_{ic}}(N_S)}{\partial \mu_i^2}$, we can obtain the equation below.

$$\frac{\partial^2 \overline{w_{ic}}(N_S)}{\partial \mu_i^2} = \frac{2s_{ic}}{(\mu_i - L_{O_i})^3}(1 + \overline{n_{ic}}(N_S - 1_c)) \tag{5.10}$$

For the convexity condition, $\frac{\partial^2 \overline{w_{ic}}(N_S)}{\partial \mu_i^2} \geq 0$, $\mu - L_{O_i}$ should be greater than zero. The constraint C2 restricts the minimum service rates VNFs. C2 includes the requirements of open and closed networks, so the convexity condition is satisfied with C2. Using the same approach, the convexity condition for open classes, $\frac{\partial^2 \overline{w_{io}}(N_S)}{\partial \mu_i^2} \geq 0$, are satisfied for all VNFs. Constraint C1 is a linear combination of $\mu_i$. The Right Hand Side (RHS) of C2 is constant. Thus, the problem is convex problem.

### 5.1.3 Heuristic Algorithm Mixed Queuing Network Model

#### 5.1.3.1 The Schweitzer Core Algorithm

The heuristic algorithm is suggested because the MVA algorithm requires a complete solution for all population in the closed network. For example, if we assume there are $N_C$ jobs in a closed class, the MVA algorithm requires the complete solution for all possible populations from $(0, 0, \ldots, 0)$ to $N_C$ about all stations, VNFs. This process demands high level of computation and storage complexity. The Schweitzer Core algorithm approximates the expected number of jobs in the closed network based on an iterative approximation technique.

Initial distribution of jobs is approximated by distributing jobs uniformly to related stations, (5.11). Since classes denote the service chains in our model, the population of closed classes will be uniformly distributed to associated VNFs depending on service chain policy.

$$\overline{n_{ik}}(N_S) = \frac{N_k}{\sum_{j \subseteq V} \underline{P}(k,j)} I(\underline{P}(k,i)), \forall k \subseteq C, i \subseteq V \tag{5.11}$$

$I(\underline{P}(c,i))$ is an indicator function, which is 0 when $\underline{P}(c,i) = 0$ and 1 when $\underline{P}(c,i)$=1 for all $c$ and $i$.

The population of $N_S - 1_c$, one c class job is reduced from $N_S$ state, state is estimated proportional to the $N_S$ state population vector in [65], where $N_C$ is the population distribution state of closed network. When one c class job is reduced from $N_S$, the expected queuing length of other classes are not changed from $N_S$ state in the first equation of (5.12). However, the expected queuing length of c class job is reduced proportional to the c class job, $N_c$ from $N_S$ state in the second equation of (5.12).

$$\begin{aligned} \overline{n_{ik}}(N_S - 1_c) = & \overline{n_{ik}}(N_S) for\ k \neq c, k \subseteq C \\ & \frac{N_c - 1}{N_c} \overline{n_{ik}}(N_S) for\ k = c, k \subseteq C \end{aligned} \tag{5.12}$$

Since $\overline{n_{ic}}(N_C)$ is approximated from (5.11), $\overline{n_{ic}}(N_C - 1_c)$ also can be estimated through (5.12). After obtaining $\overline{n_{ic}}(N_C - 1_c)$, we calculate more exact expected queuing length by using (5.2) and (5.4), $(\overline{n_{ic}}(N_S) \rightarrow \overline{n_{ic}}(N_S - 1_c) \rightarrow \overline{n_{ic}}(N_S))$. Then, we repeat this iteration until the queuing length difference between sequential iterations converges. The Core Schweitzer algorithm is described in Algorithm 4.

**Algorithm 4** The Core Schweitzer Algorithm

1: INPUT: Closed classes c, the number of VNFs V, $N_c$ for all classes c, $\overline{n_{ic}}(N_S)$ for each VNF, $s_{iC}$, and $EC_i$
2: Initialize iteration $t$ to 1.
3: **Step1.** Compute approximate $\overline{n_{ic}}(N_S - 1_c)$ from (5.12) $\forall c$.
4: **Step2.** Compute approximate $\overline{n_{ic}}(N_S)$ by using (5.3), (5.4), and $\overline{n_{ic}}(N_S - 1_c)$ $\forall k$.
5: **if** $max_{\forall i,c} \frac{(\overline{n_{ik}}(N_S)^t - \overline{n_{ic}}(N_S)^{t-1})}{N_S} \geq \frac{1}{(4000+16|N_S|)}$ **then**
6: Superscript $t$ designates iteration, then set $t = t + 1$ and goto step 1.
7: **else**
8: Go to final step.
9: Increase t to t+1.
10: **end if**
11: **Final Step.** Compute throughput estimates $\overline{x_{iC}}(N_S)$.
12: Output: $\overline{n_{ic}}(N_S)$, $\overline{x_{ic}}(N_S)$, $\overline{w_{ic}}(N_S)$, $\forall c \subseteq C$

$\overline{n_{ic}}(N_S)$ is approximated by using (5.11), which means that the number of jobs in closed queue are uniformly distributed to related service chains. Thus, we can compute the number of jobs in closed queue of $n_{ic}(N_S - 1_c)$ state by using (5.12) (line 3). Then more accurate expected queuing length of $n_{ic}(N_S)$ state is computed again by using (5.3), (5.4), and Little's law (line 4). If the expected queuing length difference between $t$ and $t+1$ iterations are greater than the evaluation factor, we increase the iteration $t$ to $t+1$ and go back to step 1. If the termination condition is satisfied, we finish the algorithm and compute the expected queuing length, throughput, and waiting time of the closed networks of $N_S$ state (line 5 - line 11).

#### 5.1.3.2 Approximate Optimal Service Rates Allocation in Mixed Queuing Network

The optimization problem could not be solved because it includes $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$. Since two terms can be obtained through recursive algorithm which requires service rates of VNFs, unknown decision variables, it has not been possible to solve the optimization problem.

We employ the initial estimation of the Schweitzer Core algorithm to estimate the expected queuing length of $\overline{n_{ic}}(N_S - 1_c)$ and $\overline{n_{ic}}(N_S)$ states. $\overline{n_{ic}}(N_S)$ is estimated by (5.11) and $\overline{n_{ic}}(N_S - 1_c)$ will be obtained by (5.12) with the estimated value of $\overline{n_{ic}}(N_S)$. By substituting $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$ in (5.4) and (5.7), the optimization problem can be solved from approximate mean queuing length of closed classes.

We propose Algorithm 5 that computes the approximate optimal service rate allocation in mixed queuing networks.

In step 1, the approximate expected queuing length of $N_S$ and $N_S - 1_c$ states are obtained in closed classes network by using (5.11) and (5.12). Iteration value k is set to 1 (line 2 - line 5). Since the $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$ are estimated by (5.11) and (5.12), we can solve the convex optimization problem that finds optimal service rate allocation by substituting $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$ in (5.4) and (5.7) in step 2 (line 6 - line 9). By using optimally allocated service rates, we can compute more exact expected queuing length by using the Schweitzer Core Algorithm. Improved expected queuing lengths $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$ are obtained through the Schweitzer Core Algorithm with inputs: $\overline{n_{ic}}(N_S)$ and $\overline{n_{ic}}(N_S - 1_c)$, and $(C_i^*)_k$

**Algorithm 5** Service Rate Allocation Algorithm in Mixed Queuing Network

**INPUT**: $s_{io}$, $s_{ic}$, $L_{Oi}$, $N_C$, $k_{max}$.
**Step1**:
  Initialize k=1.
  Obtain approximate expected queuing length of closed classes, $\overline{n_{ic}}(N_S)_k$ by using (5.11) in $N_S$ state $\forall$ $c$, $i$.
  Compute approximate expected queuing length of closed classes $N_S - 1c$ state, $\overline{n_{ic}}(N_S - 1_c)_k$ by using (5.12), $\forall$ $c$, $i$,.
**Step2**:
  Substitute $\overline{n_{ic}}(N_S)_k$ and $\overline{n_{ic}}(N_S - 1_c)_k$ in (5.4) and (5.7) and solve the optimization problem.
  Optimal service allocations, $(C_i^*)_k$, are obtained for all VNFs.
  Compute the maximum expected waiting time of service chains, $max_{(C_i^*)_k}(\underline{P}\vec{w})_k$ at $k_{th}$ iteration.
**Step3**:
  Increase the iteration k= k+1.
  Use the Schweitzer Core Algorithm with optimal service rates, $(C_i^*)_{k-1}$, to compute exact expected queuing length of closed classes.
  Obtain advanced $\overline{n_{ic}}(N_S)_k$ and $\overline{n_{ic}}(N_S - 1_c)_k$.
**Step4**:
  Solve the optimization problem by substituting $\overline{n_{ic}}(N_S)_k$ and $\overline{n_{ic}}(N_S - 1_c)_k$ in (5.5) and (5.8).
  Obtain new service rate allocation, $(C_i^*)_k$.
  Compute the maximum expected waiting time of service chain, $max_{(C_i^*)_k}(\underline{P}\vec{w})_k$
**Step5**:
**if** k=$k_{max}$ **then**
    Terminate Algorithm.
**else**
    Go back to step 3.
**end if**
**OUTPUT**: $k^* = argmin_k[max_{(C_i^*)_k}(\underline{P}\vec{w})_k]$, $(C_i^*)_{k^*}$, $max_{(C_i^*)_{k^*}}(\underline{P}\vec{w})_{k^*}$

in step 3 (line 10 - line 13). The optimization problem is initially solved with approximate expected queuing length. Therefore, the resource allocation in $k$th iteration is not optimal allocation of $k+1$th expected queuing length. In step 4, we solve the convex optimization problem with $k+1$th iteration's expected queuing length and obtain new optimal service rate allocation, $(C_i^*)_{k+1}$. Since the service rate is changed, the expected queue length of networks will be changed as well. So the algorithm goes back to step 3 and repeat the calculation of the expected queuing length and solving optimization problem (line 14 - line 17). We repeat this process until $k$ reaches $k_{max}$ iterations (line 18 - line 23). In our experiments, this algorithm could achieve convergence within 10 iterations. After finishing the algorithm, we select the optimal service rate allocation of iteration $k$ which has the minimum expected waiting time of service chains through all iterations. The complexity of algorithm can be calculated as $O(optimization\ solving\ time + (K_{max}-1)(VC^2 + optimization\ solving\ time))$.

### 5.1.4 Numerical Analysis

The proposed algorithm is implemented in MATLAB and CVX is used to solve the optimization problem. In our numerical analysis model, eight types of VNFs are assigned to a single server: NFV State Manager, Firewall, QoS, Distributed Denial of Service (DDos), WAN opt, Instruction Detection System (IDS), Load Balancer, and Rate Limiter. There are four types of service chains including different VNFs. The system model is shown in Figure 5.2.

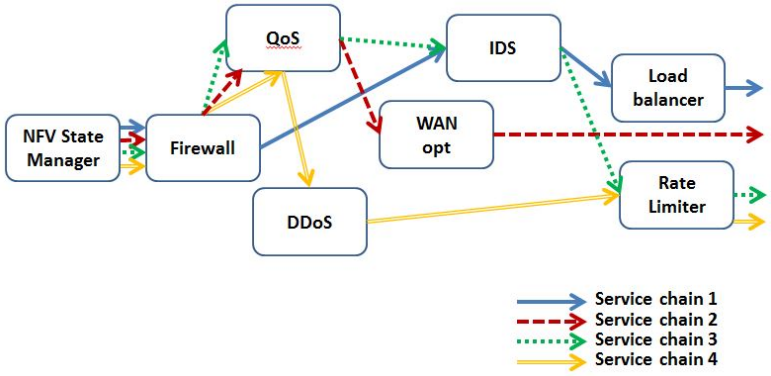


Figure 5.2: Simulation Model

Since NFV State manager works for monitoring the states of VNFs, NFV State manager is

only related to closed networks. Therefore, all closed networks require NFV State Manager but this is not included in open networks. Arrivals to each VNF are generated randomly per class according to a Poisson distribution with $\lambda = 100$. The average amount of loads per visit, $s_c$, is randomly provided by normal distribution with $\mu = 20$ and $\sigma = 5$ per classes. The number of jobs in closed networks is also randomly given by Normal distribution with $\mu = 10$ and $\sigma = 5$. The iteration parameter $k_{max}$ is set to 30 and the total service rate of the server is basically set to 60,000 per one minute.

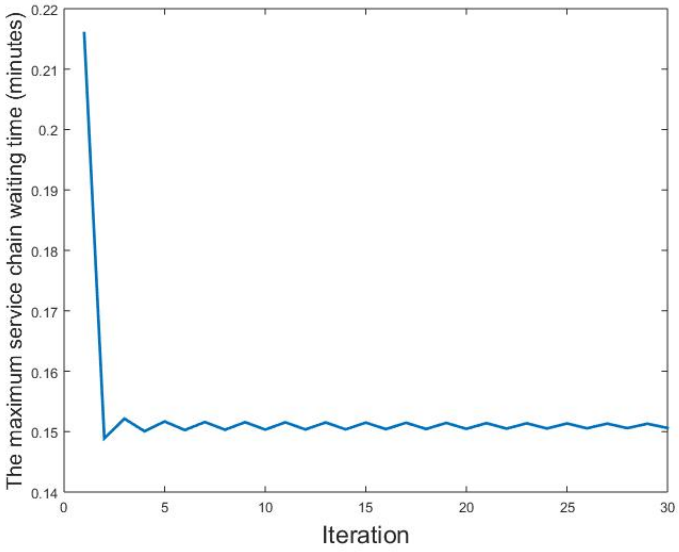


Figure 5.3: The maximum expected waiting time of service chain

Figure 5.3 shows the performance of the algorithm. As the algorithm is repeated, the expected queuing length is stabilized from the initially approximate queuing length. As iterations are repeated, the closed class queuing length is accurately computed for each VNF and service rates are optimally allocated based on these queuing length. So we could see that the expected waiting time difference between iterations becomes smaller with iterations. The Y-axis presents the maximum expected waiting time of service chains. We selected the iteration $k^*$ which has the minimum value of the objective function value and $(C_i^*)_{k^*}$ to be the approximate optimal solution.

Figure 5.4 represents the maximum expected waiting time of service chains as we increase

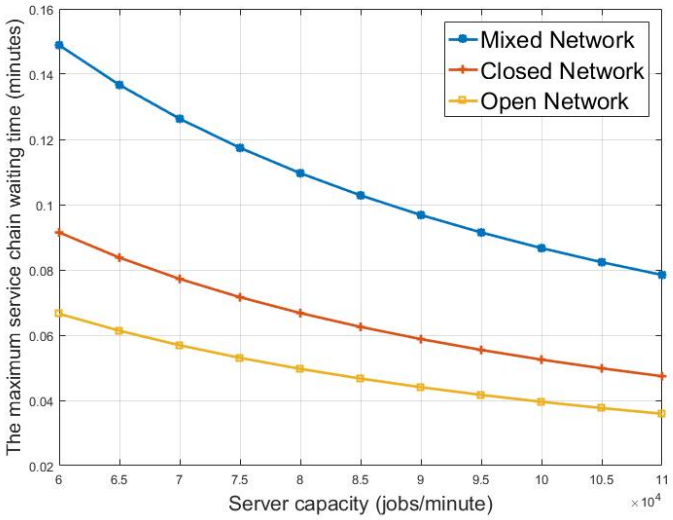


Figure 5.4: The maximum expected waiting time of service chains with increase in capacity of server

the capacity of the server. As we increase the capacity of the server, we could reduce the expected waiting time of service chains. We can observe that the closed network has more impact on waiting time of the system than the open network.

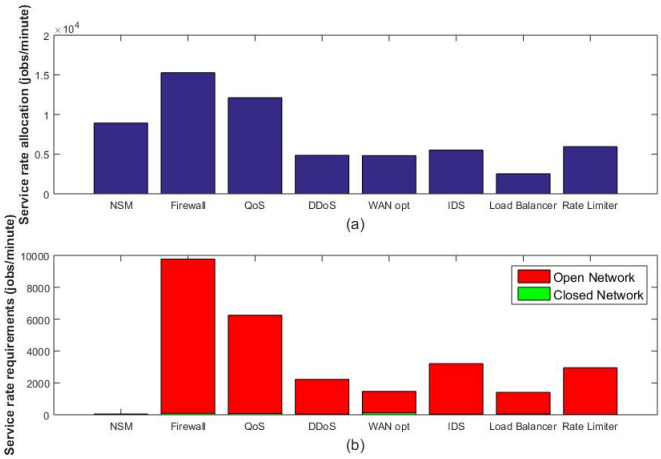


Figure 5.5: Service rate allocation to VNFs

Figure 5.5 (a) shows the approximate optimal service rate allocation to each VNF when the maximum server capacity is set to 60000. Figure 5.5 (b) presents the minimum service rate requirements of open and closed classes workloads. NFV State Manager (NSM) does not

receive any open classes loads, but the third highest service rate is assigned to NSM. Also, the high service rate is assigned to WAN opt because WAN opt receives the high workloads from the closed classes even though WAN opt receives relatively low workload by open network. Therefore, we can observe that the workloads of closed classes have a significant impact on the resource allocation of the mixed queuing network.

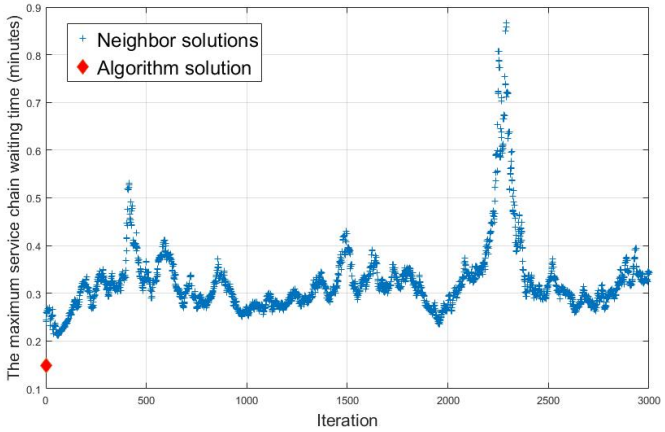


Figure 5.6: Neighbor solutions

In order to verify the effectiveness of the algorithm, we compared the approximate optimal solution with neighbor solutions. In every iteration, we search neighbor solution which has different service rate allocation and closed classes jobs distribution. Since the service rate is optimized under a given closed classes jobs distribution, it is required to change the closed class job distribution as well to find the neighbor solution. Two VNFs are randomly selected, and five percentage of service rates and the number of closed class jobs are exchanged between randomly selected VNFs. In Figure 5.6, we could observe that none of the neighbor solutions has a better objective function value than the algorithm solution.

## 5.2 NFVO and VNFM Algorithm: Power Aware NFV-SC Resource Allocation and Embedding in Hyper-scale Clouds

We consider a hyper-scale cloud infrastructure composed of multiple Software Defined Network (SDN) domains. Each domain consists of multiple data centers, SDN switches, and a

single SDN controller. The SDN controller controls the traffic flow between data centers and determines the forwarding path. The communication between domains is performed following SDN Wide Area Network (WAN) protocol. The SDN WAN controller serves as a central controller of WAN communication.

Network nodes installation is highly complex in a realistic environment. In the hyper-scale cloud infrastructure, it is extremely difficult to estimate the communication time between data centers because the forwarding path is highly dynamic which affects communication time. Unlike the traditional network environment, the routing path between nodes is determined by flow based routing strategy. This means that even the same source and destination nodes can have different routing paths depending on the network status. Therefore, a traditional Virtual Network Embedding (VNE) strategy which places virtual nodes and virtual links onto the substrate network is not an appropriate approach in SDN environment. The frequent routing path transition demands highly dynamic virtual link adjustments. Thus, it requires complex computational cost in finding optimal virtual link placement. In order to embrace the dynamic variation of the SDN environment, the probabilistic approach is utilized to estimate the expected communication time between nodes.

We employ a Stochastic Geometry technique [86] to measure the communication time between data centers. Data centers, SDN switches, BGP switches are installed based on an Poisson Point Process (PPP) and forwarding switches are selected based on an Matern Hard-core Point Process (MHPP) model using an Marked Point Process (MPP) technique. Instead of assigning virtual links onto the substrate network, we estimate the expected communication time and cost between data centers, and this estimation result affects the NFV-SC placement strategy onto the cloud infrastructure.

NFV-SCs are modeled by using an M/G/1/K queuing network. Each VNF is modeled as M/G/1/K queue and the routing probability between VNFs are periodically learned by the controller. Since the explicit analysis of the M/G/1/K queue is extremely difficult, we employ the approximation analysis technique proposed in [84], [85].

The purpose of this Section is minimizing the power consumption of the service providers' infrastructure while assuring Service Level Agreement (SLA) of NFV-SC. We formulate the

problem as a Mixed Integer Non-linear Program (MINP). However, the complexity of the problem makes it intractable to solve. So we divide the problem into two sub problems: 1) resource allocation and, 2) NFV-SC embedding. The resource allocation problem is reformulated as a convex problem so we can solve it in a reasonable time. In the resource allocation problem, we minimize the resource allocation to VNFs while guaranteeing the SLA of users in the processing level. After solving the resource allocation problem, the NFV-SC embedding problem is solved by using a heuristic algorithm based on resource allocation result. The correlation clustering machine learning technique [92] is used in the proposed heuristic algorithm so that we can assign highly correlated VNFs onto the same data center to reduce the network power.

The effectiveness of the proposed algorithm is evaluated in the numerical analysis. The system infrastructure is randomly generated in every iteration and parameter setting while fixing the total power consumption and user request levels at a similar level. After solving the resource allocation problem, we compared the performance of proposed algorithm based on the optimal solution of the resource allocation problem. The proposed algorithm is compared with Embedding Algorithm Minimizing the Power Consumption (EMPC) proposed in [97] and other traditional bin-packing algorithms. In the numerical analysis, we could observe that the proposed algorithm saves up to 15% more energy than the traditional block placement algorithm. Also, the proposed algorithm assures the SLA of users in all types of environment. The SLA violation rate usually does not exceed the 5% in most cases.

### 5.2.1 System Model

#### 5.2.1.1 SDN Cloud Computing Infrastructure

NFV-SC placement problem is considered on the hyper-scale distributed cloud computing infrastructure. Multiple Software Defined Local Area Network (SDLAN) domains are included in cloud computing infrastructures, and domains communicate through an Exterior Gateway Protocol (EGP) such as Border Gateway Protocol (BGP). Wide Area Network (WAN) communication is controlled by a Software Defined WAN (SDWAN) protocol. SDWAN is widely

studied and used because of its advantages of efficient bandwidth utilization, traffic control, and failure control. For example, Google deployed SDN principles and OpenFlow protocol to Google's WAN infrastructure and achieved highly efficient infrastructure operation [83]. Their experience shows many WAN links achieve 100% utilization and the average link utilization over 70% over long time period. This is a remarkable improvement when we consider the average utilization rate of WAN links is generally between 30 to 40%.

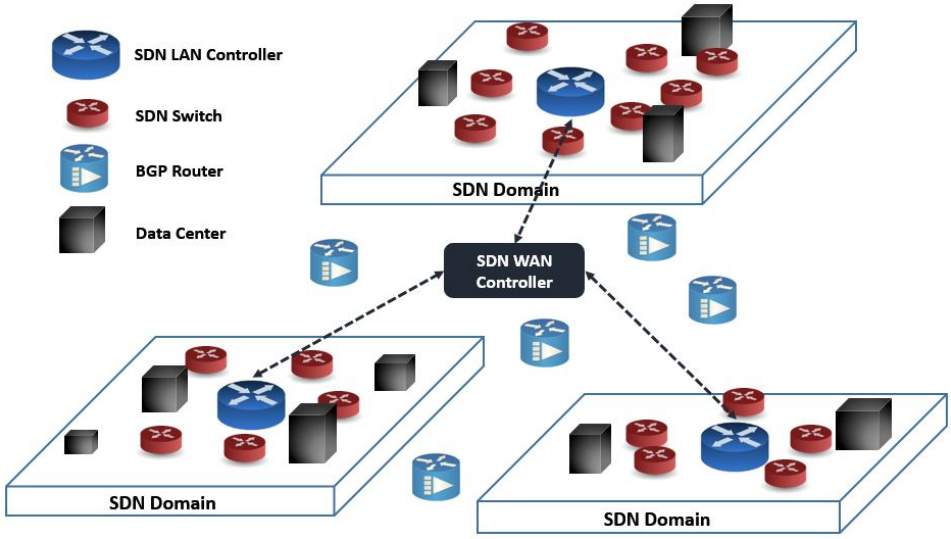


Figure 5.7: SDN cloud computing infrastructure

Fig. 1 describes our system model composed of multiple SDN domains. Each SDN service domain is controlled by an SDLAN controller. The SDLAN controller selects optimal routing paths for inter-data center communication in the same domain. When VMs request communication with a node in different domain, the SDLAN controller sends requests to the SDWAN controller, then the SDWAN controller chooses the optimal routing path for communication.

The notations used in the following formulations are described Table 5.2.

Table 5.2: Formulation Notations

| | |
|---|---|
| $V_D$ | A set of data centers |
| $E_D$ | A set of links between data centers |
| $\Phi_{DC}(A)$, $\Phi_{Switch}(A)$, $\Phi_{BGP}(A)$ | The Poisson point process of data center, switches, and BGP switches in bounded area $A$ |
| $\lambda^A_{DC}$, $\lambda^A_{Switch}$, $\lambda_{BGP}$ | The intensity of the Poisson point process $\Phi_{DC}(A)$, $\Phi_{Switch}(A)$, and $\Phi_{BGP}(A)$ in bounded area $A$. |
| $SC$ | A set of service chains. |
| $G_{SC_i}$ | Service chain graph, $SC_i \in SC$. |
| $V_{SC_i}$ | A set of Virtual Network Function (VNF) included in service chain $SC_i$. |
| $E_{SC_i}$ | A set of virtual links connecting VNFs in service chain $SC_i$. |
| $B_{p_B}(s_d^2)$ | Buffer size of queue having a squared coefficient variation of service time $s_d^2$ with blocking probability $p_B$ |
| $p_B(v)$ | The blocking probability of VNF $v$, $v \in SC_i$, $SC_i \in SC$ |
| $s_d^2$ | A squared coefficient variation of service time |
| $s_a^2$ | A squared coefficient variation of arrival time |
| $\mu(v)$ | A service rate of VNF $v$, $v \in SC_i$, $SC_i \in SC$ |
| $\widetilde{\lambda}(v)$ | Effective arrival rate to VNF $v$, $v \in SC_i$, $SC_i \in SC$ |
| $\mathcal{G}(u)$ | Gateway switch in the same domain with the data center $u$, $u \in V_D$ |
| $\mathbb{T}(u, v)$ | The communication time between node $u$ and $v$. |
| $X_{v,u}$ | Binary variable, the value is 1 when the VNF $v$ is assigned to the data center $u$. |
| $\mu_v$ | Continuous variable, the value represents the service rate assigned to VNF node $v$. |
| $\mathbb{P}^I_{server}(u)$ | The power consumption of the data center $u$ server layer in the idle state. |
| $\mathbb{P}^F_{server}(u)$ | The power consumption of the data center $u$ server layer in the full utilization state. |
| $\mathbb{P}^T_{network}(u)$ | The maximum transmission power of the data center $u$ access switch in the full bandwidth utilization state. |
| $\mathbb{P}_{switch}$ | The power consumption of the SDN switches in the full bandwidth utilization state. |
| $\Psi_{SC_i}$ | Service completion time of the service chain $i$. |
| $\mathcal{L}_{SC_i}$ | The latency constraint of the service chain $i$. |
| $C(u)$ | The processing capacity of the data center $u$. |
| $B(u)$ | The maximum transmission bandwidth of the data center $u$. |
| $B_{switch}$ | The maximum network transmission bandwidth of the SDN switches. |
| $\xi_j(SC_i)$ | User $j$th expected access time to service chain $SC_i$. |
| $M_{u,u'}$ | Traffic flow rate from the node $u$ to $u'$. |
| $R_{u,u'}$ | Routing probability from the node $u$ to $u'$. |

We model the SDN cloud computing infrastructure by means of an undirected graph, $G_D = (V_D, E_D)$. The vertex, $V_D$, expresses the data center and $E_D$ represents the connectivity between data centers. Each vertex and edge has its property as below:

- $l(v)$: the associated domain of the data center, v $\in V_D$
- $C(v)$: the maximum service rate of the data center, v $\in V_D$
- $B(e_{uv})$: the network bandwidth of links between data center $u$ and $v$, $e_{uv} \in E_D$
- $Dist(e_{uv})$: the physical distance of links between data centers $u$ and $v$, $e_{uv} \in E_D$.

For the more general analysis of the cloud computing infrastructure, we generate the cloud computing infrastructure by using the PPP in the 2-dimensional Euclidean space $R^2$ without accumulation of points. Data centers and SDN switches are generated with different intensity in each region $A$, $\lambda_{DC}^{A}$ and $\lambda_{Switch}^{A}$, selectively. BGP routers are also generated by its own intensity, $\lambda_{BGP}$, for inter-domain communication. This means that the PPP of data centers, $\Phi_{DC}$, has the following distribution:

$$P\{\Phi_{DC}(A_1) = n_1, \ldots, \Phi_{DC}(A_k) = n_k\} = \prod_{i=1}^{k} (e^{\lambda_{DC}^{A_i}} \frac{(\lambda_{DC}^{A_i})^{n_i}}{n_i!}) \tag{5.13}$$

$A_1, \ldots, A_k$ are bounded domains, which are mutually disjoint sets. Then we can measure the means of its points, data centers, by $\lambda_{DC}^{A_i}$ in each domain. In the same way, we can generate and measure the PPP of SDN switches, $\Phi_{Switch}$, and BGP routers, $\Phi_{BGP}$.

For example, we generated the SDN cloud computing infrastructure in $15000km \times 20000km$ 2-dimensional space in Fig. 5.8. Data centers are dispersed across four different SDN domains. The intensity of data centers in each domain, $\lambda_{DC}^{A}$, is randomly determined between 4 to 8. Each domain includes a single SDLAN controller, and the SDWAN controller is generated in the center of the whole domain. BGP routers are generated across all domains with intensity, $\lambda_{BGP}$, of 100. SDN switches are not displayed in the figure because of graphical complexity but they are also generated in each domain with different intensity, $\lambda_{Swich}^{A}$. Since the infrastructure is generated randomly based, it will provide more general and unbiased system performance measurements than deterministically generated infrastructure.

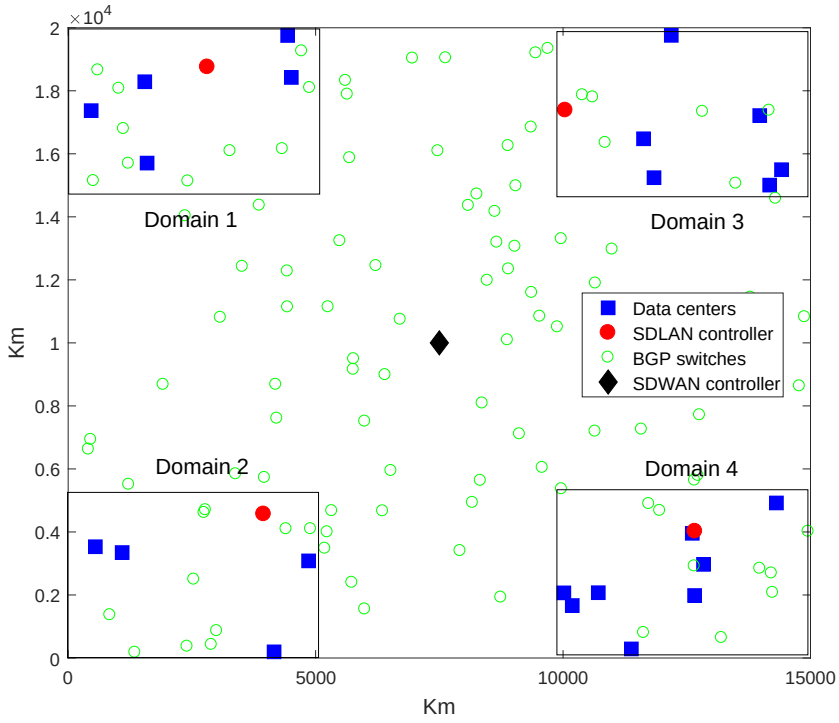


Figure 5.8: Cloud Computing Infrastructure Generated by Poisson Point Process

#### 5.2.1.2 NFV service chain modeling

NFV-SC represents a set of network functions and sequence of the request processing depending on service provider's policy. In the traditional network environment, network functions are implemented in hardware based physical devices, so the service chain was created by connecting physical devices. Thus, it was not easy to change the architecture of service chains and have flexible configuration once the service chain architecture is configured. However, NFV enables more flexible and scalable service chain configuration. Service providers can migrate VNFs to other servers or data centers depending on their objectives, or they can generate additional VNFs without additional CAPEX. Depending on the system state, service providers can change the processing sequence of jobs so that pending jobs in busy VNFs can be processed earlier in other VNFs if the sequence of the processing is not strictly forced. At a glance, the NFV-SC embedding problem looks similar with the traditional VNE problems. However, NFV-SC placement problem should be considered using the different model because it has its

unique characteristics, namely, the sequence of processing.

We employ a queuing network model for NFV-SC modeling. VNFs are assumed as independent queues and the processing sequences between VNFs are expressed in terms of the routing probabilities of edges connecting the service entrees. Routing probability is measured by network transmission history between VNFs and is periodically reported to the SDLAN controller. Since VNFs have highly different performance requirements and functions, the simple Markovian service rate queuing model is not appropriate for NFV-SC modeling. Thus, VNFs are assumed as M/G/1/K queue having different service time distributions and we will analyze this by using the approximation technique proposed in [84], [85].

Each VNF is modeled by means of an independent directed graph, $G_{SC_i} = (V_{SC_i}, E_{SC_i}), SC_i \in SC$. $G_{SC_i}$ is NFV-SC forwarding graph included in service chain $i$ which includes vertexes $V_{SC_i}$ and edges $E_{SC_i}$. Each vertex and edge has its property as below:

- $\lambda_{ext}(v)$: external job arrival rates to the VNF $v$, $v \in V_{SC_i}$, $SC_i \in SC$.
- $R_{uv}$: the routing probability from VNF $u$ to $v$, $\forall\ u, v \in V_{SC_i}$, $\forall\ SC_i \in SC$.

The closed form expressions of M/G/1/K systems are very difficult to be obtained and its complexity makes it difficult to be used in many applications. So we use the approximation approach proposed in [84], which provides a closed-form expression for the system performance measurements. They used a two-moment approximation technique that approximates the performance measurements of M/G/1/K queue from the M/M/1/K formula.

$$B_{p_B}(s_d^2) = B_{p_B}(1) + NINT\{\frac{1}{2}(s_d^2 - 1)\sqrt{\rho}B_{p_B}(1)\} \tag{5.14}$$

where $NINT$ is the nearest integer, $B_{p_B}(s_d^2)$ is buffer capacity of the system excluding those in service for a particular threshold blocking probability value $p_B$ with a squared coefficient of variation of service $s_d^2$, $\rho$ represents the utilization rate of the queue, $\rho = \frac{\lambda}{\mu}$, $\lambda$ dedicates the arrival rate to the queue, and $\mu$ means the service rate of the queue. This means that the weighted combination of the buffer size of Markovian systems is an effective strategy to approximate the buffer size of the $M/G/1/K$ queue.

By using this approximation, we can find the approximated buffer size of the M/G/1/K queue from closed form expression of the M/M/1/K formula. The M/M/1/K formula can be described as below:

$$B_{p_B}(1) = \frac{ln(p_B/(1-\rho+p_B\rho))}{ln(\rho)} \tag{5.15}$$

where $B_{p_B}(1)$ is buffer capacity of the system in Markovian service rate queue with the blocking probability $p_B$. From (5.15) and (5.14), the buffer size of the M/G/1/K queue with blocking probability $p_B$.

$$\begin{aligned} B_{p_B}(s_d^2) &= (\frac{ln(p_B/(1-\rho+p_B\rho))}{ln(\rho)} - 1) + \frac{1}{2}(s_d^2-1)\sqrt{\rho}(\frac{ln(p_B/(1-\rho+p_B\rho))}{ln(\rho)} - 1) \\ &= \frac{(ln(p_B/(1-\rho+p_B\rho)) - ln(\rho))(2+\sqrt{\rho}s_d^2-\sqrt{\rho})}{2ln(\rho)} \end{aligned} \tag{5.16}$$

If we invert the last expression, we can express the blocking probability, $p_B$, of the M/G/1/K queue as:

$$p_B = \frac{\rho^{(\frac{\sqrt{\rho}s_d^2-\sqrt{\rho}+2K}{2+\sqrt{\rho}s^2-\sqrt{\rho}})}(\rho-1)}{\rho^{2(\frac{\sqrt{\rho}s_d^2-\sqrt{\rho}+K+1}{2+\sqrt{\rho}s^2-\sqrt{\rho}})}-1} \tag{5.17}$$

The expected number of jobs in M/M/1/K systems is described as below:

$$L_{M/M/1/K} = \frac{\rho}{1-\rho} - \frac{(B_{p_B}(M)+1)\rho^{B_{p_B}(M)+1}}{1-\rho^{(B_{p_B}(M)+1)}} \tag{5.18}$$

If we substitute $B_{p_B}(1)$ by (5.15), then we can describe (5.18) as below:

$$L_{M/M/1/K} = \frac{\rho(ln(1/\rho) + p_B ln(p_B/(1-\rho+p_B\rho)) + p_B ln(\rho))}{ln(1/\rho)(\rho-1)} \tag{5.19}$$

By using the two-moment, and matching (5.19), we can derive the expected queuing length of M/G/1/K queue.

$$L_{M/G/1/K} = \frac{\rho(\ln\frac{1}{\rho} + p_B\ln(\frac{p_B}{1-\rho+p_B\rho}) + p_B\ln(\rho))(2+\sqrt{\frac{\rho}{e}}s_d^2 - \sqrt{\frac{\rho}{e}})}{2\ln\rho(\rho-1)} \tag{5.20}$$

This approximation equation is accurate when $s_d^2 \geq 1$. This means that we assume, that the VNFs' service time has high variance, so the minimum service variation is equal to 1, the Markovian service rate.

In the resource allocation problem, we will assign a sufficient storage for the queue to lower the blocking probability and make it close to zero. Therefore, the influence of $p_B$ is a minor

factor in the expected queuing length of M/G/1/K queue. So we can simplify equation(5.20) by assuming $p_B \approx 0$. Therefore, equation (5.20) can be approximately simplified by assuming $p_B \approx 0$ as follows,

$$\begin{aligned} L_{M/G/1/K} &= \frac{\rho(\ln\frac{1}{\rho})(2+\sqrt{\frac{\rho}{e}}s_d^2-\sqrt{\frac{\rho}{e}})}{2\ln\rho(\rho-1)} \\ &= \frac{\rho(\ln\frac{1}{\rho})(2+\sqrt{\frac{\rho}{e}}(s_d^2-1))}{2\ln\rho(\rho-1)} \end{aligned} \tag{5.21}$$

**Proposition 1** Convexity of queuing length function:

*The approximated expected queuing length of M/G/1/K queue $L_{M/G/1/K}$ is a convex function in terms of its service rate $\mu$*

**Proof** $L_{M/G/1/K}$ can be expanded by replacing $\rho$ to $\frac{\lambda}{\mu}$.

$$\begin{aligned} L_{M/G/1/K} &= -\frac{\rho(2+\sqrt{\frac{\rho}{e^{s_d^2}}}(s_d^2-1))}{2(\rho-1)} \\ &= \frac{\rho}{1-\rho}+\frac{1}{2}\frac{s_d^2-1}{\sqrt{e^{s_d^2}}}\sqrt{\rho}(\frac{\rho}{1-\rho}) \\ &= \frac{\lambda}{\mu-\lambda}+\frac{1}{2}\frac{s_d^2-1}{\sqrt{e^{s_d^2}}}\sqrt{\frac{\lambda}{\mu}}(\frac{\lambda}{\mu-\lambda}) \end{aligned} \tag{5.22}$$

The first term of (5.22) is a convex function in terms of the service rate $\mu$. If we can prove that the second term of (5.22) is convex, the (5.22) is a convex function. In order to prove the convexity of the function, the second derivative of the second term, $\overline{L}$, should be greater than zero. We assume the squared coefficient of service time $s_d$ is a constant. □

The expected waiting time of the M/G/1/K queue can be described as below by using Little's law and expanding (5.21).

$$\begin{aligned} W_{M/G/1/K} &= \frac{L_{M/G/1/K}}{\lambda} \\ &= \lambda\{\frac{\lambda}{\mu-\lambda}+\frac{1}{2}\frac{s_d^2-1}{\sqrt{e}}\sqrt{\frac{\lambda}{\mu}}(\frac{\lambda}{\mu-\lambda})\} \\ &= \frac{1}{\mu-\lambda}+\frac{1}{2}\frac{s_d^2-1}{\lambda\sqrt{e}}\sqrt{\frac{\lambda}{\mu}}(\frac{1}{\mu-\lambda}) \end{aligned} \tag{5.23}$$

#### 5.2.1.3 Matern Hardcore Point Process (MHPP)

SDN separates the control and data planes, which enables centralized control of the network infrastructure. Although SDN takes advantage of centralized control in terms of optimized forwarding path selection, optimal resource utilization, low delay, and so on, the transition from source-destination based forwarding strategy to flow-based forwarding strategy makes it difficult to estimate the networking performance in SDN-based network infrastructures. The routing path is highly dynamic in SDN network infrastructure and varies depending on the link state of switches.

Thus, the probabilistic modeling can be an applicable solution for the performance analysis in SDN network infrastructure. We employ an MHPP for SDN network analysis. The MHPP forms a point process from the PPP and MPP. In the MHPP, points are never closer to each other than some given distance, $h_{min} > 0$. Underlying point process is generated by the PPP and certain points are removed from the PPP depending on the mark attached to each point. In other words, we generate SDN data plane switches based on the PPP, and select some switches for packet forwarding depending on its mark in every distance. A mark can be any characteristic of switches such as utilization rate, incoming traffic, or service rate. We assume that the SDN controller makes the optimal decision for path selection by selecting SDN switches having lowest utilization rate in its neighboring area. So selected switches cannot be closer than the distance $h_{min}$ for forwarding delay minimization in switches, and farther than $h_{max}$ for signal strength. For the communication between two nodes having a distance $r$, SDN controller selects switches in the circle boundary which has the radius of $\frac{r}{2}$ by having two nodes on its edge.

SDN switches are generated by the PPP with intensity $\lambda^{A}_{Switch}$ on $R^2$ in region $A$.

$$\Phi_{Switch}(A) = \sum_{i \in L} \epsilon_{x_i} \tag{5.24}$$

where $\epsilon_{x_i}$ is measure on the bounded Euclidean space $A$ at a point $x_i$.

The distance between SDN switches is controlled by the intensity, $\lambda^{A}_{Switch}$ as described in Proposition 2.

**Proposition 2** The distance between points

*The distance between points does not exceed $h_{max}$, if the points are generated by PPP with the intensity $\lambda^A_{Switch} = -\frac{ln(1-\alpha)}{\pi h^2_{max}}$ where $\alpha$ is the high probability close to 1 and $h_{max}$ is the maximum distance between points.*

**Proof** When underlying points are generated by a PPP, the distance to the nearest neighbor point can be expected by using a void probability, the probability that no points are existing in the area. If we assume the distance from a point $x$ to its nearest point is $h_x$, the probability that nearest point exists in the distance $h_{max}$ from the point $x$ is

$$P(h_x \geq h_{max}) = P(\Phi_{Switch}(B(x, h_{max})) = 0) = e^{-\lambda^A_{Switch}\pi h^2_{max}} \tag{5.25}$$

where $B(x, h_{max})$ is the closed circle centered at $x$.

In other words, if we want the distance between points not to exceed $h_{max}$, we can find the probability,

$$P(h_x < h_{max}) = 1 - P(h_x \geq h_{max}) = \alpha \tag{5.26}$$

where $\alpha$ is high probability that is close to 1. From (5.25) and (5.26), we can find the equation below.

$$e^{-\lambda^A_{Switch}\pi h^2_{max}} = 1 - \alpha \tag{5.27}$$

If we take the log on both sides, then

$$\lambda^A_{Switch} = -\frac{ln(1-\alpha)}{\pi h^2_{min}} \tag{5.28}$$

This means that if the intensity of the PPP, $\lambda^A_{Switch}$, is equal to (5.28), the distance between points generated by PPP does not exceed $h_{max}$ with very high probability, $\alpha$. □

Marks are independently generated for each point, and it represents the utilization rates of switches. Since the switches are used for SDN traffic and traditional Ethernet packet forwarding as well, we independently generate the utilization rates of a point, $U_i$, and its values are randomly generated between lower bound of utilization rate, 0, and upper bound utilization rate, 1. So the following MPP is generated with the set of utilization rate, $U_i$ of each switch.

$$\widetilde{\Phi}_{Switch}(A) = \sum_{i \in A} \epsilon_{(x_i, U_{x_i})} \tag{5.29}$$

Based on this, we generate a new mark $m_i$ of each points as follows,

$$m_i = \mathbb{1}(U_{x_i} < U_j, \quad \forall x_j \in A_{x_i}(h_{min}) \backslash x_i) \tag{5.30}$$

$A_{x_i}(h_{min})$ represents the circle area having radius $h_{min}$ and its origin of $x_i$. $\mathbb{1}(x)$ is the indicator function. So the new mark $m_i$ is the indicator that shows point $x_i$ as the switch having the lowest utilization rate among all the points in its neighborhood $A_{x_i}(h_{min})$. Since we select the lowest utilization rate SDN switch for forwarding, the mark $m_i$ also indicates that if this SDN switch is selected for flow forwarding or not.

Then MHPP is defined by

$$\Phi_{MHPP}(A) = \sum_{i \in l} m_i \epsilon_{(x_i, U_{x_i})} \tag{5.31}$$

We can identify the distribution of MHPP by first finding the distribution of the mark of $\Phi_{MHPP}(A)$, and then calculating the intensity $\lambda_{MHPP}$ of $\Phi_{MHPP}$ as described in [86]. For the bounded area $A$, and $0 \leq U_{x_i} \leq 1$ by Slivnyak's theorem [86].

$$\begin{aligned}
&\widetilde{C}(A \times ([0,1] \times 1)) \\
&= E[\int_A \int_{[0,1]} \mathbb{1}(U_{x_i} < U_j \quad \forall x_j \in A_x(h_{min}) \cap \Phi \backslash \{x\}) \cdot \widetilde{\Phi}_{Swtich}(d(x, U_{x_i}))] \\
&= \lambda^A_{Switch} |A| \int_A \int_0^1 P\{(\sum_{(x_j, U_j) \in \widetilde{\Phi}_{Swtich}} 1(U_j \leq u) \epsilon_{x_j}) \cdot (A_x(h_{min})) = 0\} du dx \\
&= \lambda^A_{Switch} |A| \int_0^1 e^{-\lambda^A_{Switch} u v_d h^d_{min}} du \\
&= |A| \frac{1 - e^{-\lambda^A_{Switch} v_d h^d_{min}}}{v_d h^d_{min}}
\end{aligned} \tag{5.32}$$

where $\widetilde{C}(\cdot)$ indicates Campbell measure, which gives the expected number of points in a given space. $v_d$ represents the volume of the ball in d dimension, $v_d = \sqrt{\pi^d}/\Gamma(1 + d/1)$. We consider the distribution in the plane so $d$ is equal to 2, and $v_d = \pi$. Thus, we can find the intensity of selected SDN switches for packet forwarding by using MHPP depending on the distance between data centers u and v, $Dist(e_{uv})$.

$$\lambda_{MHPP}(Dist(e_{uv})) = |A(Dist(e_{uv}))| \frac{1 - e^{-\lambda^A \pi h^d_{min}}}{\pi h^d_{min}} \tag{5.33}$$

The MHPP guarantees that distance between points, selected switches, cannot be closer than $h_{min}$. Since the distance between overlay points does not exceed the $h_{max}$ and the MHPP selected points not closer than $h_{min}$, the MHPP has the distribution of points having a distance between $h_{min}$ and $h_{max}$.

### 5.2.2 Problem Formulation

NFV-SCs are modeled as directed graphs which have multiple vertices and edges. Since the NFV-SCs will be embedded onto the distributed cloud computing infrastructure, the expected NFV-SC completion time includes the expected processing time in VNFs, communication time between VNFs, and user access time to NFV-SC. This section introduces a way to obtain the expected processing time in VNFs by using M/G/1/K queuing network theory, and the expected communication time between data centers by using MHPP and M/G/1 queue analysis so that we can calculate the communication time between VNFs depending on its allocation onto the data centers. The infrastructure power minimization NFV-SC resource allocation and embedding problem will be formulated by Mixed Integer Nonlinear Problem (MINP).

#### 5.2.2.1 Service Processing Time

VNFs are modeled as M/G/1/K queue, so NFV-SCs are represented as multiple independent M/G/1/K queuing networks. Explicit analysis of M/G/1/K queuing network is extremely difficult because the service time does not follow Markovian property. So we employ the approximated M/G/1/K queuing network analysis approach introduced in [85]. The expressions for the M/G/1/K queue approximation is derived in Section 5.2.1.2. In M/G/1/K queuing network analysis, J.M.Smith [85] used the expansion method that utilizes the blocking probability of the queue. The expansion method has followed three stages: 1) Network reconfiguration, 2) Parameter estimation, 3) Feedback elimination.

In the first step, the queuing network is expanded by adding an artificial node, M/M/$\infty$ queue, for holding the jobs when the M/G/1/K queue is saturated as described in Fig. 5.9. The blocked jobs proceed to abstraction queue a with probability $p_B$ and next node N2 with probability $1 - p_B$.

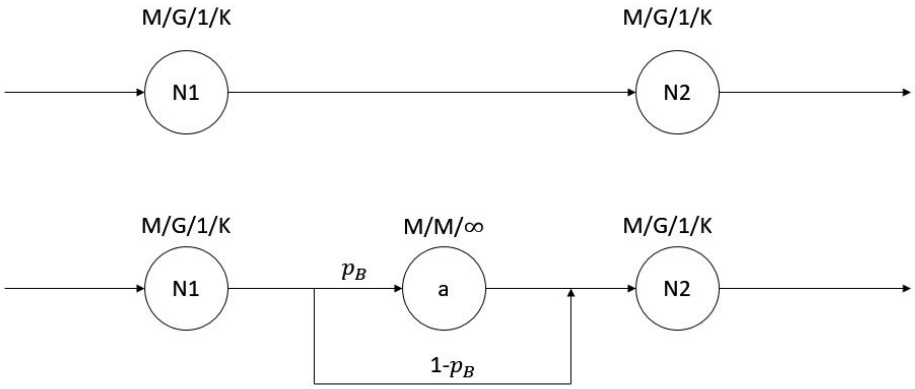


Figure 5.9: M/G/1/K queuing network expansion method analysis

In the second step, parameters, efficient arrival rates to queue, feedback blocking probability of the abstract queue, and service rate of abstract queue are estimated caused by the influence of the abstract queue. The approximation is finalized by removing the feedback loop of abstraction queue.

However, our assumption is that the storage size assigned to each M/G/1/K queue, VNFs, is sufficient so that the blocking probability $p_B$ become close to zero. In other words, we can ignore the blocking probability of M/G/1/K queue, so the abstraction node does not need to be added. The M/G/1/K queuing network can be analyzed as the connection of independent M/G/1/K queues when the blocking probability is close to zero.

Unlike Markovian queues, the output process of each queue is not Poisson process. Thus, the arrival rates to subsequent queues will not follow Poisson arrivals either. To account for the general arrival processes occurring in the network, we employ the Allen-Cuneen approximation [87]. The waiting time in the queue depends on the squared coefficient of variation of the arrival process, $s_a^2$, and service process, $s_d^2$. We assume that these squared coefficients of variation are periodically learned through system monitoring. By using (5.23), we can find the expected waiting time of each queue in the queuing network as follows.

$$W_q = \sqrt{\frac{(s_a^2 + s_d^2)}{2}} W_{M/G/1/K} \tag{5.34}$$

We restate the expected waiting time of the VNF $v$ as below,

$$W_q(v) = \sqrt{\frac{(s_a^2(v) + s_d^2(v))}{2}}[\frac{1}{\mu(v) - \widetilde{\lambda}(v)} + \frac{1}{2}\frac{s_d^2(v) - 1}{\widetilde{\lambda}(v)\sqrt{e}}\sqrt{\frac{\widetilde{\lambda}(v)}{\mu(v)}}(\frac{1}{\mu(v) - \widetilde{\lambda}(v)})] \tag{5.35}$$

Each node has different service rate, $\mu(v)$, squared coefficient of variation of arrival process, $s_a^2(v)$, and service process, $s_d^2(v)$, and the efficient arrival rate, $\widetilde{\lambda}(v)$. The arrival rates to node v can be expressed as $\widetilde{\lambda}(v) = \lambda_{ext}(v) + \sum_{v' \in SC_i} R_{v'v}\widetilde{\lambda}(v')$.

Therefore, the expected waiting time of the NFV-SC, $SC_i$, can be computed by using (5.35).

$$W_{SC_i} = \sum_{v \in SC_i} W_q(v) \tag{5.36}$$

#### 5.2.2.2 Communication Time between Data Centers

The inter-data center communication is known a-priori to the controller for NFV-SC placement. Since the routing paths are highly dynamic even between the same source and destination data centers, the expected communication time is estimated by using a probabilistic approach. The communication time between data centers is measured depending on the physical distance and the SDN domains that the data centers belong to. The expected number of SDN forwarding switches for communication are estimated by using the MHPP introduced in Section 3.3 and switch ports are modeled as M/G/1 queues. We assume that switches are independent of each other. Thus, the computation of expected waiting time in switches are obtained through independent queue analysis. The approximated expected waiting time of the M/G/1 queue is derived using the well-known Pollaczek Khintchine approximation formula [116].

$$E[W_{M/G/1}(s)] = \frac{s_d^2(s) + 1}{2} E[W_{M/M/1}(s)] \tag{5.37}$$

where $s_d^2(s)$ is the squared coefficient of variation of service time which is measured in SDN switches $s$. The service rate of switch $s$ is $\mu(s)$ and arrival rates are uniformly generated between $[\lambda_l(s), \lambda_h(s)]$. So the expected waiting time in each SDN switch is derived as,

$$
\begin{aligned}
E[W_{switch}(s)] &= \int_{\lambda_l(s)}^{\lambda_h(s)} \frac{(s_d^2(s)+1)}{2} \frac{1}{\mu(s)-\lambda(s)} p(\lambda(s)) d\lambda(s) \\
&= \int_{\lambda_l(s)}^{\lambda_h(s)} \frac{(s_d^2(s)+1)}{2} \frac{1}{\mu(s)-\lambda(s)} \frac{1}{\lambda_h(s)-\lambda_l(s)} d\lambda(s) \\
&= \frac{(s_d^2(s)+1)}{2(\lambda_h(s)-\lambda_l(s))} [-log(\mu(s)-\lambda(s))]_{\lambda_l(s)}^{\lambda_h(s)} \\
&= \frac{(s_d^2(s)+1)}{2(\lambda_h(s)-\lambda_l(s))} log(\frac{\mu(s)-\lambda_l(s)}{\mu(s)-\lambda_h(s)})
\end{aligned}
\tag{5.38}
$$

where $p(\lambda(s))$ represents the probability density function (pdf) of the random variable $\lambda(s)$. So we can compute the expected waiting time in each switch using (5.38).

The SDN controllers are also assumed as M/G/1 queues. Therefore, we can find the expected waiting time in SDN controllers $c$, $E[W_{controller}(c)]$, using the same approach in (5.38) but with different service rates, arrival rate, and squared coefficient of variation parameters.

The communication time between data centers are measured depending on how data centers are geographically distributed.

**1) Intra-data center communication time**

When VNFs are assigned to the same data center, we assume the communication cost is equal to zero. Technically, the communication delay exists for intra-data center communication depending on data center architecture and VNF placement. However, this delay has a minor influence on the service completion time in hyper-scale distributed data center infrastructure.

$$
\mathbb{T}(u,v) = 0, \qquad u = v \tag{5.39}
$$

$\mathbb{T}(u,v)$ is the communication time measurement function that returns the communication time between nodes $u$ and $v$.

**2) Inter-data center communication time in the same SDN domain**

Inter-data center communication in the same SDN domain is implemented based on the path selection of the SDLAN controller. The SDLAN controller establishes the packet forwarding path between data centers depending on the network status.

$$
\mathbb{T}(u,v) = \lambda_{MHPP}(Dist(e_{uv}))E[W_{Switch}]+E[W_{Lcont}]+E[D_{prop}(u,v)], \qquad u \neq v \quad and \quad l(u) = l(v) \tag{5.40}
$$

The first term explains the expected waiting time in SDN switches for flow forwarding depending on the distance between nodes $u$ and $v$. $E[W_{Lcont}]$ describes path calculation and processing time spent in SDLAN controller, and the propagation delay, $D_{prop}$, depends on the physical distance between data centers as described in (5.40). The propagation delay, $D_{prop}$, can be any monotonically increasing random variable depending on the distance.

**3) Inter-data center communication time in different SDN domains**

When data centers are located in disparate domains, this causes higher communication delay. The infrastructure is composed of heterogeneous SDN domains controlled by a single controller and controllers communicates with each other by using an exterior gateway protocol such as the Broder Gateway Protocol (BGP). Each SDLAN controller communicates through BGP routers and this is controlled by an SDWAN controller. Therefore, the inter-data center communication time in different regions is described as (5.41).

$$\mathbb{T}(u,v) = \mathbb{T}(u,\mathcal{G}(u)) + \mathbb{T}(\mathcal{G}(u),\mathcal{G}(v)) + \mathbb{T}(\mathcal{G}(v),v), \qquad u \neq v \quad and \quad l(u) \neq l(v) \tag{5.41}$$

$\mathcal{G}(u)$ represents the gateway switch located in the same SDN domain with node $u$. The data center node $u$ should transmit the packet through the gateway switch for the inter-data center communication. Thus, the communication time between the source data center and gateway switch is measured in the same manner as the communication time between nodes in the same domain (5.40). Also, the transmission time between gateway switches is measured in the same approach because WAN communication follows SDWAN controllers path selection strategy. The last term indicates the communication time between destination gateway to destination data center with the same strategy.

#### 5.2.2.3 Power Aware NFV-SC Resource Allocation and Embedding

In this section, we introduce the optimal service rate allocation to VNFs and NFV-SC placement problem by using Mixed Integer Non-linear Programming (MINP). The objective of the problem is minimizing the power consumption of the cloud infrastructure used for NFV-SCs processing. The optimization problem determines the service rates of VNFs and distribution of VNF node to data centers while guaranteeing the QoS of the service chain.

The optimization problem is modeled as an MINP with multiple constraints which guarantees SLA of NFV-SC and capacity constraints of data centers. The MINP is given by:

$$\begin{aligned}
&\underset{X_{v,u},\mu_v}{\text{minimize}} \sum_{u \in V_D} [\mathbb{P}_{server}(u) + \mathbb{P}_{network}(u)] \\
&\text{subject to} \\
&\quad C1 : E[\Psi_i] \leq \mathcal{L}_{SC_i} \quad \forall SC_i \in SC \\
&\quad C2 : \sum_{SC_i \in SC} \sum_{v \in V_{SC_i}} \mu_v X_{v,u} \leq C(u), \quad \forall u \in V_D \\
&\quad C3 : \sum_{u \in V_D} X_{v,u} = 1, \quad \forall v \in V_{SC_i}, SC_i \in SC \\
&\quad C4 : X_{v,u} = \{0,1\}, \quad \forall v \in V_{SC_i}, SC_i \in SC, u \in V_D \\
&\quad C5 : \mu_v \geq \widetilde{\lambda}(v), \quad \forall v \in V_{SC_i}, SC_i \in SC
\end{aligned} \tag{5.42}$$

The objective function minimizes the total power consumption of the data center server layer for NFV service processing, and the cloud network infrastructure including data center network devices for traffic transmission and SDN switches used for flow forwarding. Many researchers studied the data center energy consumption model in the wide area in terms of processor power, memory, storage, and data center level models [88]. For the server layer power consumption estimation of data centers, we employ the model that measures the power consumption of the data center based on the server utilization rate [89], [90].

$$\mathbb{P}_{server}(u) = \mathbb{P}^I_{server}(u) + (\mathbb{P}^F_{server}(u) - \mathbb{P}^I_{server}(u)) f_P(U(u)) \tag{5.43}$$

where $U(u) = \frac{\sum_{v \in V_{SC_i}, i \in SC} \mu_v X_{v,u}}{C(u)}$ is the utilization rate of the data center $u$. $U(u)$ is obtained by dividing service rates assigned to VNFs in the data center $u$ by the capacity of the data center $C(u)$. $f_P(U(u)) = U(u)^\eta$ is the penalty function which has its range between 0 to 1 as the utilization rate of the data center $u$. If we want to give more penalty for highly utilized data centers, we can set the high value of the penalty parameter $\eta$ so that the difference between low utilization rate and high utilization rate have a large gap in terms of its power consumption. In (5.43), the data center server layer basically spends $\mathbb{P}^I_{server}$ even in the idle state to keep the activation state, and the power consumption increases as the utilization rate of the data center

increases by the penalty function $f_p(U(u))$ within the range of [ $\mathbb{P}^I_{server}$ $\mathbb{P}^F_{server}$].

The network transmission power is measured depending on the network bandwidth utilization rate of the infrastructure.

$$\mathbb{P}_{network}(u) = \mathbb{P}^T_{network}(u)\frac{\sum_{u' \in V_D, u' \neq u} M_{u,u'}}{B(u)} + \sum_{\substack{u' \in V_D \\ u' \neq u}} \lambda_{MHPP}(Dist(e_{u,u'}))\frac{M_{u,u'}}{B_{switch}}\mathbb{P}_{switch} \quad (5.44)$$

$M_{u,u'}$ is the total traffic flow rate generated from the source data center $u$ to destination data center $u'$ in (5.44).

$$M_{u,u'} = \sum_{SC_i \in SC} \sum_{\substack{v \in SC_i \\ v' \in SC_i}} \widetilde{\lambda}_v X_{v,u} R_{v,v'} X_{v',u'} \quad (5.45)$$

The first term of (5.44) describes the traffic flow transmission power consumption spent by the data center $u$. The second term expresses the power spent by SDN switches for transmitting jobs to other data centers. We assume SDN switches have homogeneous condition, so they consume the same power for transmitting the same flow rate. The power consumption of SDN switches is calculated by multiplying the expected number of forwarding switches estimated by MHPP, $\lambda_{MHPP}(Dist(e_{u,u'}))$, and the proportional power consumed by a single switch for transmitting flow rate out of the maximum bandwidth capacity of the switch, $(M_{u,u'}/B_{switch})\mathbb{P}_{switch}$.

The first constraint $C1$ restricts NFV-SC completion time not to exceed the service chain latency for SLA. NFV-SCs have heterogeneous latency constraints due to their different purposes. The delay-sensitive service chain will have a tight latency constraint, but delay-tolerant service chain will have a loose latency constraint relatively.

$E[\Psi_i]$ includes expected users' access time to target NFV-SC, processing time in VNFs, and the communication time between VNFs in the same NFV-SC.

$$E[\Psi_{SC_i}] = max_j(\xi_j(SC_i)) + \sum_{v \in SC_i} W_q(v) + \sum_{\substack{u \in V_D \\ u' \in V_D}} M_{u,u'}\mathbb{T}(u,u') \quad \forall SC_i \in SC \quad (5.46)$$

The multiple user requests are randomly generated from VMs located in dispersed data centers. Thus, user access time to NFV-SC is affected based on how the service chains are distributed and placed in distributed data center infrastructure. So users will have a different expected access time to target NFV-SC. For fair service of all users, the maximum access time of users is included to measure the service chain completion time. The processing time in

VNFs are calculated by the expected waiting time in M/G/1/K queuing networks proposed in Section 4.1. The expected communication time between VNFs are computed based on the routing probability and network flow transmission time between VNFs.

The constraint $C2$ explains that the total service rates of VNFs assigned to the each data center cannot exceed the capacity of the data center. The constraint $C3$ represents the VNFs should be assigned to a single data center. $X_{v,u}$ is defined as a binary decision variable in $C4$. For the stability of the system, the service rate of VNFs should be greater than the effective arrival rate to VNFs in $C5$.

The problem includes multiple multiplications of integer variables and continuous variables or integer variable and integer variables. The multiplication of integer variable with continuous and integer variables and multiplication of integer variable and integer variable can be linearized by a simple substitution technique described in [9].

### 5.2.3 NFV-SC Resource Allocation and Embedding Algorithm

The joint optimization of the service rate allocation and NFV-SC embedding is an NP-hard problem because solving the VNE problem by means of MILP is a well known NP-hard problem for general virtual network and substrate network graph [94]. Thus, we propose a two-step approach that divide the joint problem to two sub-problems: NFV-SC resource allocation and NFV-SC embedding problems. In the first step, we solve the NFV-SC resource allocation problem by formulating the problem by convex problem. Based on the resource allocation result, the proposed heuristic algorithm achieves NFV-SC embedding by using a correlation clustering unsupervised learning algorithm.

In order to guarantee the SLA of NFV-SC, we split the latency constraint of each service chain to processing latency, $\mathcal{L}^{p}_{SC_i}$, and communication latency, $\mathcal{L}^{c}_{SC_i}$.

#### 5.2.3.1 VNF Service Rate Allocation

The service rate allocation problem is described as a convex problem.

$$\underset{\mu_v}{\text{minimize}} \sum_{\substack{v \in V_{SC_i} \\ SC_i \in SC}} \mu_v$$

subject to

$$\begin{aligned} C1 &: \sum_{v \in SC_i} W_q(v) \leq \mathcal{L}^p_{SC_i} \quad \forall SC_i \in SC \\ C2 &: \sum_{\substack{v \in SC_i \\ SC_i \in SC}} \mu_v \leq \sum_{u \in V_D} C(u) \\ C3 &: \mu_v \geq \widetilde{\lambda_v} \quad \forall v \in SC_i, SC_i \in SC \end{aligned} \tag{5.47}$$

Since the server layer power consumption of data center increases as the utilization rate of the data center increases, the service rate allocation problem assigns the minimum the service rate to VNFs while guaranteeing the SLA of the NFV-SC. The objective function is the linear summation of service rates of VNFs. The first constraint $C1$ forces the expected waiting time of each NFV-SC not to exceed the processing latency constraint. We proved that the expected waiting time in M/G/1/K queue is a convex function in Appendix A. The Left-Hand Side (LHS) of the constraint $C1$ is a summation of convex functions, so the constraint $C1$ is convex. The total service rate allocation of VNFs cannot exceed the maximum capacity of the cloud infrastructure in $C2$. Also, the $C3$ defines the service rate of each VNFs to be greater than the effective arrival rate to VNFs for stability of the system. The optimal service rate of VNFs can be obtained by solving (5.47) while assuring the NFV-SC processing delay constraint is satisfied.

#### 5.2.3.2 Service Chain Embedding Algorithm with Correlation Clustering (SCEACC)

The service rates of VNFs are determined by solving the optimization problem which minimizes the total utilization rate of the cloud infrastructure while assuring SLA. Then, we need to make a decision of VNFs block placement that minimizes the power consumption of the infrastructure consumed for NFV services.

The network layer power consumption is caused by the inter-data center communication. In other words, if all VNFs are assigned to a single data center, it will be the best solution to reduce the network power consumption. However, the isolated assignment of VNFs causes a high user access time for distant users, hence the service provider may not be able to offer a fair service to all users. Also, the data center capacity constraint will restrict the isolated allocation of VNFs. Consequently, a suitable distribution of VNFs is required so that the service providers reduce the power consumption of the data center while assuring fair service quality to all users.

We propose a heuristic algorithm that searches the efficient NFV-SC embedding strategy using a correlation clustering machine learning technique. Through correlation clustering, VNFs having a high traffic transmission rate are clustered into the same cluster and assigned to the same data center in order to reduce the inter-data center communication traffic.

#### 5.2.3.3 Correlation Clustering Unsupervised Learning

Correlation clustering is one of the unsupervised machine learning techniques that provides a method for clustering sets of a vertices in a graph in order to achieve the optimum number of clusters without a priori knowledge of the number of clusters. The relationship between objects is represented as a graph with vertices and edges. Each vertex represents an object and edges show the correlation between objects. Positive edges stand for the similarity between objects and the negative edges express the dissimilarity between objects. The purpose of the clustering is maximizing agreements (sum of positive edge weights within a cluster plus the absolute value of the sum of negative edge weights between clusters) or minimizing disagreements (absolute value of the sum of negative edge weights within a cluster plus the sum of positive edge weights across clusters). Unlike other clustering algorithms, correlation clustering does not require a priori knowledge of the number of clusters. The clustering result finds the optimal number of clusters that optimizes the objective function.

In NFV-SC clustering, the purpose is minimizing the network traffic flow between clusters so that we can reduce the network transmission power through the clustering. Thus, we select the objective function which minimizes disagreements of clusters.

NFV-SC is modeled as a directed graph, $G_{SC_i} = (V_{SC_i}, E_{SC_i}) \quad \forall i \in SC$, and its edges

show the routing probability between VNFs. However, the edges of the service chain model do not include the negative edges which show the dissimilarity of VNFs. So, we need to transform the service chain graph so that the graph can express the similarity and dissimilarity between VNFs.

In the first step of the graph transformation, we obtain the average routing probability of each pairs of VNFs and transform the graph to the undirected graph, $\widetilde{G}_{SC_i} = (\widetilde{V}_{SC_i}, \widetilde{E}_{SC_i})$, where $\widetilde{R}_{uv} = (R_{uv} + R_{vu})/2, \quad \forall e \in E_{SC_i}, SC_i \in SC$. In the second transformation, the negative edges are generated by subtracting an evaluation factor, $\delta$, from routing probabilities of graph. $\widetilde{R}_{u,v} = \widetilde{R}_{u,v} - \delta$. The evaluation factor is a criteria value that distinguishes between the correlation of the VNFs. If the average routing probability between node $u$ and $v$ is greater than the evaluation factor, the edge will have a positive value and they are assumed as correlated VNFs. If the average routing probability between vertex $u$ and $v$ is less than the evaluation factor, the edge will have a negative value and they are assumed uncorrelated VNFs. Based on the transformed graph $\widetilde{G}_{SC_i}$, we can implement correlation clustering using the Integer Programming (IP) below. The set $\mathcal{E}_{SC_i}$ includes the edges having a positive routing probability from the transformed graph, $\widetilde{G}_{SC_i}$.

$$\begin{aligned} &\underset{d_{uv}}{\text{minimize}} \sum_{i \in SC} [\sum_{e \in \mathcal{E}_{SC_i}} \widetilde{R}_{uv} d_{uv} + \sum_{e \notin \mathcal{E}_{SC_i}} \widetilde{R}_{uv}(1 - d_{uv})] \\ &\text{subject to} \\ &\quad C1 : d_{uv} = \{0, 1\} \quad \forall u, v \in SC_i, SC_i \in SC \\ &\quad C2 : d_{uw} \leq d_{uv} + d_{vw} \quad \forall u, v \in SC_i, SC_i \in SC \end{aligned} \tag{5.48}$$

$d_{uv}$ is a binary variable that has its value 0 when the vertices $u$ and $v$ are in the same cluster and 1 when the vertices $u$ and $v$ does not belong to the same cluster. The first term of the objective function is the summation of weights of edges having the positive value and the second term is the summation of the negative value edges weights. The objective function stands for the correlation value between clusters. In order to minimize the objective function, the problem will cluster VNFs having positive value together, and separates VNFs having negative edges values.

However, this problem is known to be NP-complete problem, so we use the relaxation based near optimal approximation algorithm proposed in [92] to solve the clustering problem.

After clustering, we obtain a set of clusters $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$ and the number of clusters $k$.

#### 5.2.3.4 Service Chain Embedding Algorithm with Correlation Clustering (SCEACC)

We propose a heuristic algorithm which finds the minimum power consumption service chain placement by utilizing the correlation algorithm.

**Algorithm 6** SCEACC Algorithm

1: **INPUT**: $G_D = (V_D, E_D)$, $\widetilde{C}(V_D)$, $\widetilde{B}(E_D)$, $\mathbb{T}(u,v)$ $\forall u, v \in V_D$, $G_{SC_i} = (V_{SC_i}, E_{SC_i})$, $\widetilde{\lambda}(V_{SC_i})$, $W_q(V_{SC_i})$, $\mu_{V_{SC_i}}$, $\forall SC_i \in SC$, $\delta$.
2: Transform the NFV-SC graphs $G_{SC_i}$ to $\widetilde{G}_{SC_i}$, $\forall i \in SC$ by the evaluation factor $\delta$.
3: Cluster $\widetilde{G}_{SC_i}$ using (5.48). Obtain sets of clusters $\mathcal{F}$ and the number of clusters $k$.
4: $\hat{\mathcal{F}} = \mathcal{F}$
5: **while** $\hat{\mathcal{F}} \neq \emptyset$ **do**
6: Sort the clusters $\mathcal{F}$ in decreasing order of its size, $\hat{\mathcal{F}} = sort(\mathcal{F})$
7: **for** $i = 1 : k$ **do**
8: $\mathcal{A}(\hat{\mathcal{F}}_i, V_j)$ and calculate $\mathbb{P}_T(\hat{\mathcal{F}}_i, V_j)$ $\forall V_j \in V_D$.
9: Select the data center, $V_j$, which has the minimum power consumption when we assign the cluster $\mathcal{F}_i$.
10: **if** $E[\Psi_i] \leq \mathcal{L}^c_{SC_i}$ and $\sum_{v \in \hat{\mathcal{F}}_j} \mu_v \leq \widetilde{C}(V_j)$ are satisfied for the $\mathcal{A}(\hat{\mathcal{F}}_i, V_j)$ **then**
11: Assign $\mathcal{F}_i$ to $V_j$
12: Extract $\mathcal{F}_i$ from $\mathcal{F}$
13: **else**
14: Keep $\mathcal{F}_i$ to $\mathcal{F}$
15: **end if**
16: **end for**
17: $\delta = \gamma\delta$
18: Repeat the clustering (5.48) about the unassigned cluster, $\mathcal{F}$, with the updated evaluation value and find the number of clusters $k$.
19:
20: **end while**
21: **OUTPUT**: $X_{v,u}, \forall v \in V_{SC_i}$, $SC_i \in SC$, $\forall u \in V_{DC}$.

We have knowledge of the infrastructure graph, $G_D = (V_D, E_D)$, the instant capacity of data centers, $\widetilde{C}(V_D)$, the instant available bandwidth of data centers, $\widetilde{B}(E_D)$, the communication time between all data centers, $\mathbb{T}(u,v)$, the VNF service chain graphs, $G_{SC_i} = (V_{SC_i})$, the effective arriving rates to VNFs, $\widetilde{\lambda}(V_{SC_i})$, and the waiting time in VNFs, $W_q(V_{SC_i})$, the assigned service rates of VNFs obtained from (5.47), $\mu_{V_{SC_i}}$.

In the first step, we need to transform the graph as introduced in the previous section with the evaluation factor $\delta$ (line 2). After transforming the graph, the clustering is implemented

for service chain graphs using the clustering algorithm. Based on the clustering results, the number of clusters, $k$, and sets of clusters are acquired, $\mathcal{F}$ (line 3). Each cluster includes VNFs having a high correlation. In the last step, we find the data center which has the lowest power consumption for the cluster assignments while satisfying the constraint proposed in (5.42).

In order to place large size clusters first, we sort the clusters based on their service rate requirements (line 5). Then we check the availability and the infrastructure power consumption for the placements. Each cluster is assigned to all data centers and we heuristically search the total power consumption of the infrastructure , $\mathbb{P}_T$, for all assignment cases (line 7 - line 8). Then, we select the data center that consumes the minimum power when we assign the cluster $\mathcal{F}_i$ (line 9 - line 11). If the cluster is assigned to the data center, we remove the cluster from the sets $\mathcal{F}_i$. If the cluster is not assigned to data center because the latency condition and the data center capacity condition is not satisfied, we keep the cluster $\mathcal{F}_i$ in the set (line 13). If the set $\mathcal{F}$ is not empty, this means that there are still unassigned VNFs because of violation of the capacity or latency constraints. To satisfy the constraints, the clusters are separated into smaller pieces based on higher evaluation factor $\delta$. The evaluation factor increases in every iteration with the same ratio and the correlation clustering is applied for unassigned VNFs. Then unassigned clusters are separated into smaller pieces and search data centers satisfying its capacity and latency constraints while minimizing power consumption of the infrastructure(line 17). Unassigned clusters check the availability for placement with the same strategy (line 6 - line 15). If we find the appropriate data center for the cluster placement, the clusters are embedded on the data centers. This process is repeated until all VNFs are placed on the distributed cloud infrastructure. As a result of the algorithm, the embedding of the VNFs is determined, $X_{v,u}$.

### 5.2.4 Numerical Analysis

We used MATLAB to implement the SCEACC algorithm and CVX to solve the optimization problems (5.47) and (5.48) with MOSEK solver. The effectiveness of the proposed algorithm is evaluated in the randomly generated infrastructure and service chain models.

V. Persico *et al* intensively investigated the network performance of public cloud data centers

around the world in extreme hyper-scales [117]. Four different geographical locations of data centers are included in the investigation: Europe (EU), North Virginia (US), South America (SA), and Asia-Pacific (AP). They exploited two service providers cloud infrastructures, Amazon Web Services (AWS), and Microsoft Azure and measured practical network performance of inter-data center communication between each region. The distance between data centers varies between 5000 km to 16000 km. According to their analysis of AWS infrastructure, SA, EU, and US are included in the same domain and AP is separated in an independent domain with others. So the inter-data center communication between SA, EU, and US has a low Round Trip Time (RTT) and the low number of traversed hops is comparable to the inter-data center communication with AP area. The number of traversed hops are measured depending on the distance between data centers and we could infer that the packet should traverse one hop per 771.32 km on the average.

Based on this practical analysis of public cloud infrastructures, we build our simulation model. The cloud infrastructure is built on geographically distributed SDN domains. We generated four different SDN domains in a $15000 \times 20000$ km$^2$ area as shown in Fig. 5.8 and each domain has $5000 \times 5000$ km$^2$ coverage. Since the infrastructure installation and NFV-SC requests are randomly generated in every iteration, the analysis results are measured by means of repeated simulations' results.

The data centers are generated according to the PPP with an intensity $\lambda_{DC}^{A}$ between 4 to 8 in each region $A$. The data center capacity is an environment variable, which represents the currently available service rates of data centers. The data center capacity is varied through the simulation and we observed how it affects simulation results. Since the data center are not only used for NFV service, we assume that some resources of the data centers is in use by other services. Thus the utilization rate of data centers is randomly given between 0.35 to 0.45. The connectivity between data centers is randomly made within the same domain and each link has a 10 Gbps bandwidth.

The network infrastructure is composed of multiple SDN switches and BGP routers for inter-data center communication. We assume that install the maximum distance between SDN switches does not exceed 50km. SDN switches are installed following the PPP and its intensity,

$\lambda^A_{Switch}$, is determined according to Theorem 2. The distance between BGP switches is also restricted to 100 km and the intensity of the installation, $\lambda^A_{BGP}$, is achieved in the same way. Each domain has SDLAN controller and we assume that is located in the center of the domain. SDWAN controller is placed in the center of the whole domain. We select the SDN switches for flow forwarding and the distance between selected switches cannot be closer than 500 km. For the inter-domain communication, BGP routers are selected in the same strategy by SDN WAN controller. The selected BGP routers cannot be closer than 700 km.

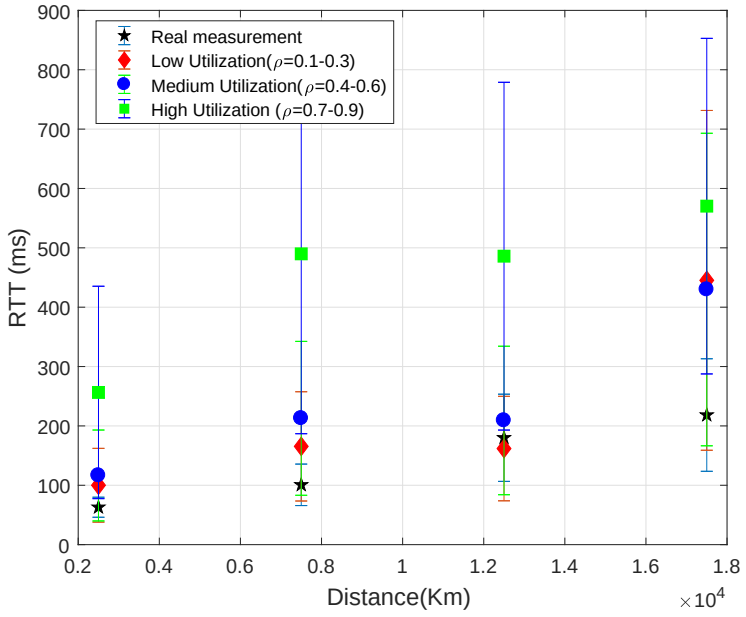


Figure 5.10: Round Trip Time comparison(RTT) with practical measurement and estimated value using proposed approach

Fig. 5.10 compares the RTT of inter-data center communication estimated by our proposed approach in Section 4.2 using MHPP and practical measurement investigated in [117]. The simulation model randomly generates following the PPP with the intensity of 6 and the distance between them dynamically varies between 0 to 20000 km. We divided the RTT measurement range to four ranges depending on the distance between data centers: 0-5000 km, 5000-10000km, 10000-15000km, 15000-20000km. The RTT is collected in each range and the mean RTT and 95% confidence interval is obtained. The first range represents the inter-domain data center communication RTT because a single domain has 5000 km by 5000 km size. Since a packet

traverses the same domain, we can observe that the confidence interval is narrower than other ranges. The estimated RTT by the proposed approach shows a similar value with a practical measurement of RTT. When the utilization state of the network is set to low and medium, the estimated RTT has only 10 to 20 ms difference from practical RTT measurements. Since high levels of network utilization is not common, we can conclude that the proposed approach finds highly accurate communication time estimation.

In order to estimate the expected waiting time in each switch and router, we need to know the service rate, arrival rate, and squared coefficients of service variation, $s_d^2$ of them. We assume a homogeneous SDN switch and BGP router infrastructure, so an equivalent service rate is assigned to all SDN switches, $\mu_{switch} = 1000$ jobs/sec and $\mu_{BGP} = 5000$ jobs/sec. Since the SDN switches and BGP routers are not only used for NFV service, the arrival rates are randomly determined while assuring stability of the devices.

We create five heterogeneous NFV service chains. Each service chain includes 6 to 7 VNFs and external arrival rates to each VNF are randomly distributed. Since service chains require different levels of SLA, the latency constraint, $\mathcal{L}_{SC_i}$, is randomly generated between 1.5 to 2 seconds for each service chain. The routing probability between VNFs is also arbitrarily determined. The squared coefficients of service time variation of VNFs are basically set to 1, which is equivalent to Markovian service rate. We will observe how the squared coefficients of variation of service rate affect the system performance with the variation of its value.

The idle state power consumption of data centers, $\mathbb{P}^I_{server}$, is set between 10000 and 15000W, and the full utilization state power consumption of data centers, $\mathbb{P}^F_{server}$, is set between 20000 and 25000W. The data center consumes power for the network, $\mathbb{P}^B_{network}$, depending on the number of its access points because the data center connection is randomly generated. Each access point consumes 472W when it uses full bandwidth for data transmission. Each SDN switch and BGP router, $\mathbb{P}_{switch}$, consumes 250W in the maximum bandwidth utilization state.

A hundred users are created in the simulation. Users' requests occur from the VM arbitrarily embedded in the distributed data centers. Users' access probabilities to VNFs of the service chain are randomly decided, so user access time, $\xi_j(SC_i)$, to the service chain is determined based on the NFV-SC embedding strategy on the distributed data centers.

In the algorithm implementation, the latency constraint of service chains is separated to processing latency and communication latency constraint as below.

$$\mathcal{L}_{SC_i} = \beta_1 \mathcal{L}^p_{SC_i} + \beta_2 \mathcal{L}^c_{SC_i} \tag{5.49}$$

The sum of $\beta_1$ and $\beta_2$ is equal to 1. We evenly distribute the latency constraint by setting the parameter $\beta_1$ and $\beta_2$ to 0.5.

The simulation is repeated 50 times for each parameters setting, then we obtain mean values of the power consumption, the service chain completion time, SLA violation, and the number of clusters in each scenario. The evaluation factor $\delta$ is set to 0.15 for the clustering and the evaluation factor increases 5% in every iteration in order to cluster failed clusters.

The SCEACC algorithm performance is compared with the four different allocation disciplines. The service rate of VNFs are determined by solving (5.47), and then the embedding performance is compared with other disciplines listed below.

· **Embedding Algorithm Minimizing the Power Consumption (EMPC)**: The power consumption minimizing virtual network embedding algorithm is proposed in [97]. The algorithm considers the energy efficiency of the substrate node one by one in increasing order and assigned VMs to high energy efficient nodes. Two conditions, the node and link availability, are verified in terms of bandwidth in the allocation process.

· **Random Allocation (RA)**: Randomly place the VNF block to data center which has the available capacity.

· **First fit Bin-Packing (FBP)**: Sort data centers in decreasing order in their available capacity levels, and place VNFs sequentially from the first data center while not violating the capacity constraint.

· **Best fit Bin-Packing (BBP)**: Place a VNF to the data center which has the lowest available capacity and greater than VNF capacity.

Fig. 5.11 shows the total infrastructure power consumption as the maximum and minimum capacity of each data center increase. In each scenario, the maximum and minimum capacity

of each data center is increased 10%. The blue line dedicates the total power consumption of the SCEACC algorithm. According to the results, the proposed algorithm saves 13.67% power consumption compare to the FBP, 9.448% less than EMPC algorithm, 7.98% less than RA, and 2.76% less than BBP. All algorithms placed the same sizes of VNFs because the size of VNFs are determined by (5.47). Thus the power saving of Fig. 5.11 is achieved by the network traffic minimization and the efficient data center utilization rate management. Fig. 5.12 represents the network power consumption of the infrastructure.

Since the BBP algorithm places the VNFs in highly utilized state data centers, it can consolidates the VNFs to the limited data centers. Thus, the network power consumption is lower than other traditional algorithms. When the capacity of the data center increases, we can observe that the network power consumption of the BBP is lower than the proposed algorithm because BBP consolidates most of the VNFs in limited data centers.

The proposed algorithm exhibits lower network power consumption than other algorithms in most cases. The SCEACC algorithm consumed approximately 45% lower network power than other algorithms except for BBP for the described reason. Although BBP does not consume high level of power for the network, it consumes the highest server power because the data center utilization rate soars with VNFs consolidation.

Since the penalty function charges higher server layer power consumption in high utilization state of the data center as we can see in (5.43). Thus, it consumes more server layer power as we can see in Fig. 5.13 when the maximum capacity of the data center is low. FBP also embeds VNFs to the high utilization state data center first. Thus, we can observe that FBP consumes highest server power than other algorithms. As we can see, the proposed algorithm manages the appropriate utilization rate of data centers through efficient VNF placement for power saving.

Since EMPC algorithm considers power efficiency of the data centers mainly, it could reduce a significant server layer power consumption. However, it causes the high level of network power consumption. So the total power consumption of the EMPC algorithm performs worse than the SCEACC algorithm.

Fig. 5.14 shows the expected service completion time of NFV-SC. The latency constraints

of the NFV-SCs are generated between 1.5 to 2 seconds as described. If we observe Fig. 5.14, we can observe that the average service completion time is lower than the 1.8 seconds in most cases. The processing time of VNFs is equivalent for all strategies because the same service rates are allocated for all strategies. Therefore, the service completion time has a difference in communication time between VNFs and user access time. As the data center capacity increases, FBP and BBP consolidate the VNFs to limited data centers. So, we can observe that the gap between these algorithms and the SCEACC algorithm is reduced in the high capacity state.

Fig. 5.15 describes how the service completion time affects user SLA violation. We generated a hundred users across the whole data centers. Since a user accesses a single NFV-SC and service completion time includes the user access time, processing time, and communication time, the NFV-SC completion time is inconsistent depending on users' VM location and NFV-SC embedding. If NFV-SC completion time of the user exceeds the latency constraint of the requested service chain, we define it for SLA violation. RA and EMPC have a high SLA violation rate because the VNFs embedding is arbitrarily achieved and EMPC only considers the power efficiency in VNF embedding while satisfying the bandwidth constraint. The proposed algorithm has lower than 8% SLA violation rate in all situations. As the capacity of the data center increases, we have more flexibility of service chain embedding. Thus, we could reduce the SLA violation as well. FBP and BBP have a pretty high SLA violation rate in tight data center capacity environments. FBP has around 20% and BBP has 10% SLA violation rates. However, the SLA violation rate decreases as the capacity of the data center increases and achieves less than 5% violation rate. Thus, we can conclude that VNFs consolidation strategy is also an effective method when the data center capacity is sufficient but it has a disadvantage in busy data center state.

Fig. 5.16 represents the number of clusters generated by the proposed algorithm as the capacity of data centers increases. As the capacity increases, the possibility of the cluster rejection is reduced. Therefore, the number of clusters are decreased in Fig. 5.16, which means more VNFs are embedded in the same data centers.

In the second scenario, the simulation is run by varying the squared coefficient of variation of service time, $s_d^2$, of VNFs. The data center capacity is fixed to 30000 jobs/sec and we observed

how the $s_d^2$ of VNFs influence on the performance and power consumption of the infrastructure. The $s_d^2$ initially is set to 1 which is the same as Markovian service rate, and is increased by 0.2 in every iteration. The high value of $s_d^2$ means service time for a single job processing highly fluctuates. Therefore, it demands high resource allocation of VNFs. Also, the expected throughput of VNFs raises, and it causes the elevation of network traffic between VNFs. In Fig. 5.17, the overall power consumption of the infrastructure increases with the increase of $s_d^2$. The power consumption of the proposed algorithm soars when the $s_d^2$ exceeds 1.6 and it is greater than other algorithms. The service chain embedding is performed by recognizing the service chain completion time in the proposed algorithm. Although we can observe that the SCEACC algorithm spends more power than other algorithms, we can still guarantee the satisfaction SLA of users through the latency aware service chain embedding. In Fig. 5.19, the SLA violation rate of FBP and BBP is between 10 to 20% and it surges as $s_d^2$ increases. However, the proposed algorithm keeps the SLA violation rate below 10% except for highly variable environments.

The last scenario presents the importance of the evaluation factor selection, $\delta$. The capacity of the data center is fixed to 20000 and the $s_d^2$ is set to 1, which is the Markovian service rate. If the evaluation factor $\delta$ is too low, we assume the most of the VNFs are correlated and obtain positive edges. Thus, it causes very large size clusters. In contrast, if $\delta$ is too high, most of edges are transformed to negative edges, which leads to the separation of VNFs. In Fig. 5.20, traditional algorithms have a similar power consumption through the iterations because they are not affected by the evaluation factor. The small variation is caused because of randomly generated parameters. However, the power consumption of the proposed algorithm is affected by adjustment of $\delta$. When $\delta$ is too low, the proposed algorithm shows worse performance than other algorithms. This is because low $\delta$ generates large size clusters. Since the experiment is implemented on high data capacity utilization environment, it causes the re-clustering of clusters. The SCEACC algorithm has the best performance result when $\delta$ is between 0.1 and 0.15. It has the lowest power consumption of the infrastructure and the number of clusters. The low number of clusters means high levels of VNFs consolidation and network traffic reduction. Thus it could reduce the power consumption because the network power consumption is

reduced as well. High $\delta$ does not achieve any clustering advantage. We generated around 30 VNFs in the simulation, and if the number of clusters is close to 30, it means that the clustering is not achieved for any of VNFs. So the proposed algorithm shows worse energy saving performance than other algorithms because the algorithm decides the embedding strategy by taking care of the latency constraint. The SLA violation rate does not exceed 25% with adjustment of $\delta$. Although the BBP has the lower SLA violation rate, it consumes much higher power consumption than the proposed algorithm.

## 5.3 Chapter Summary

We propose an efficient NFV MANO algorithm in this Chapter.

In the first Section of Chapter, VIM algorithm is proposed that maximize the performance of NFV-SC in the limited resource. NFV service chains are modeled as a BCMP mixed queuing network and formulated a convex problem to minimize the expected waiting time of service chains. Since the problem can not be solved in the mixed queuing network because the expected queuing length cannot be calculated without knowledge of service rate, an approximate service rate allocation algorithm is proposed. In numerical analysis, we could observe that closed class networks have more influence on the expected waiting time of the system. So high service rates are assigned to VNFs receiving heavy requests from closed class networks.

The joint problem of NFV-SC resource allocation and embedding in hyper-scale cloud computing infrastructure was studied in the second section of this Chapter. The problem was modeled as MINP problem in order to minimize the power consumption of the infrastructure while assuring SLA satisfaction of users. However, the complexity of the problem made it difficult to solve the problem so we proposed the SCEACC algorithm to find the efficient embedding strategy after assigning resources by using convex problem. The proposed algorithm exhibited the overwhelming embedding performance in terms of power saving and user SLA assurance in any environment of the infrastructure. Based on the numerical analysis result, we could find that efficient consolidation of VNFs should be considered in VNF-SC embedding. Too much consolidation of VNFs increased the data center utilization rate, so it caused server layer power increase and an unfair users SLAs. On the other hand, dispersed placement of the

VNFs raised the network transmission power consumption and time. This Chapter has studied off-line problem of NFV-SC resource allocation and embedding problem. For the future work of proposed approach, we expect to expand this problem to efficient NFV-SC reconfiguration and migration strategy as network status and user requirement changes over the time.

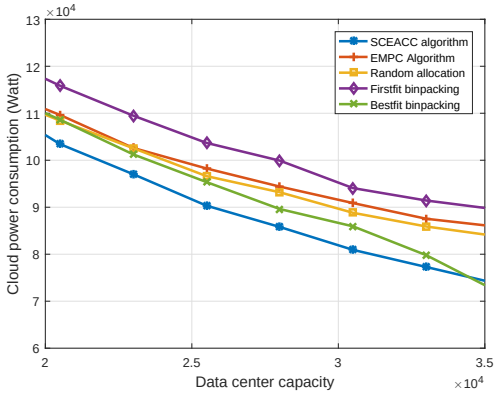


Figure 5.11: Total power consumption of the infrastructure

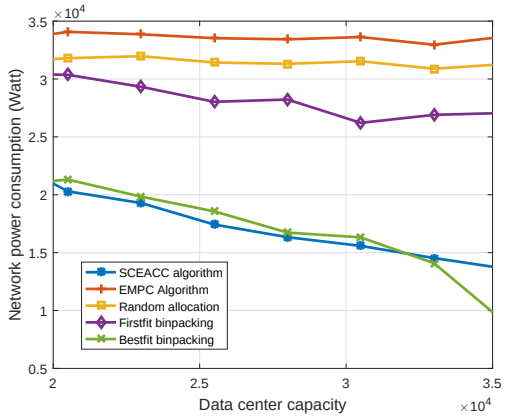


Figure 5.12: Network power consumption of the infrastructure

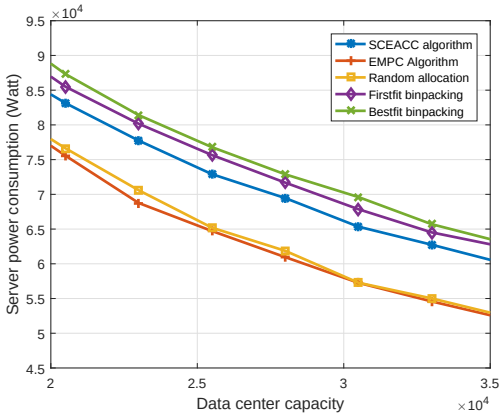


Figure 5.13: Server power consumption of the infrastructure

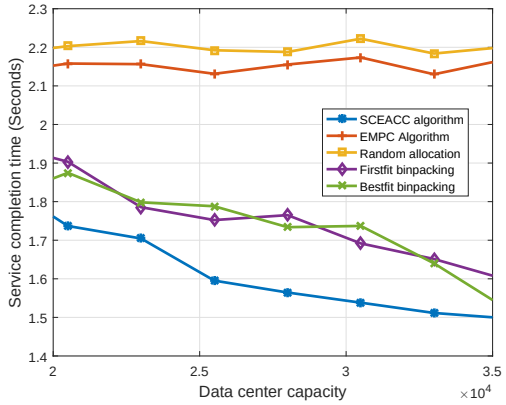


Figure 5.14: Average service chain completion time

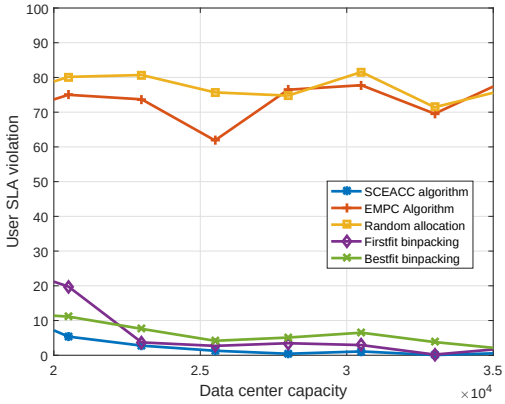


Figure 5.15: Users' SLA violation rate

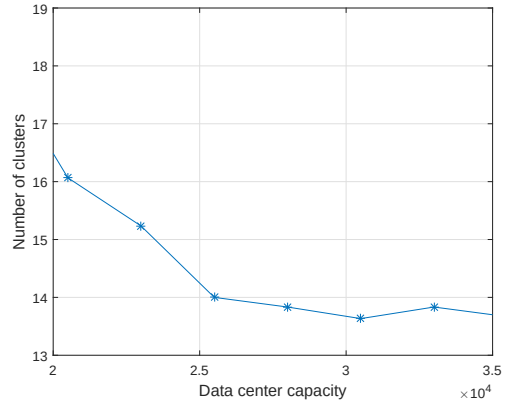


Figure 5.16: The number of cluster

**Scenario 1**: The maximum capacity of the data center $C(u)$ is adjusted from 20000 to 35000, the squared coefficient of variation of service time $s_d^2$ is set to 1, and the evaluation factor $\delta$ is 0.15.

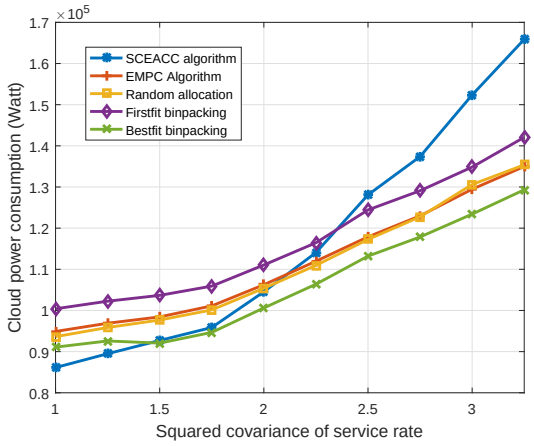


Figure 5.17: Total power consumption of the infrastructure

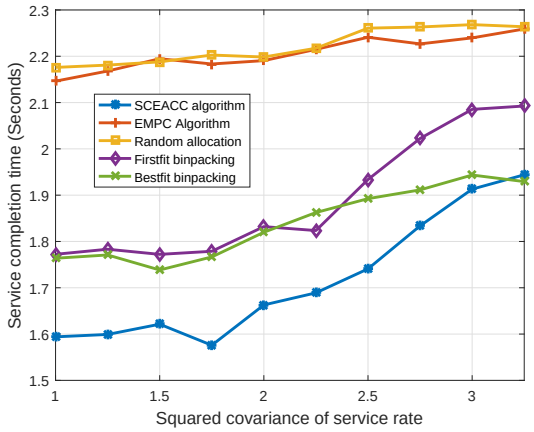


Figure 5.18: Average service chain completion time

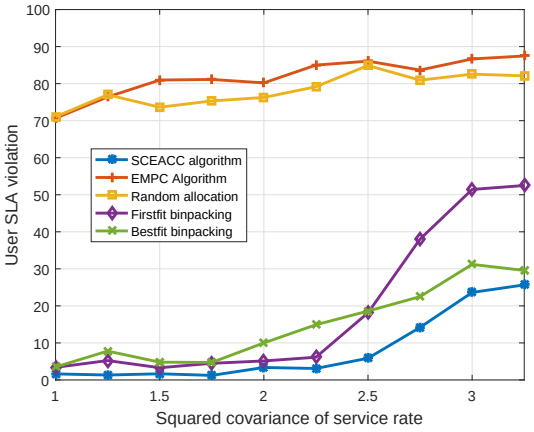


Figure 5.19: Users' SLA violation rate

**Scenario 2**: The squared coefficient of variation of service time $s_d^2$ is adjusted from 1 to 3.5, the maximum capacity of the data center $C(u)$ is set to 30000, and the evaluation factor $\delta$ is 0.15.

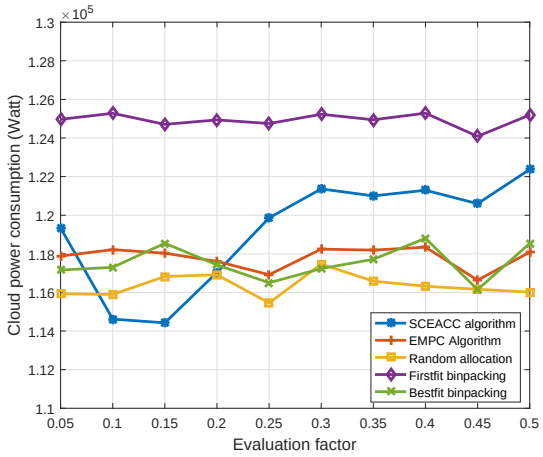


Figure 5.20: Total power consumption of the infrastructure

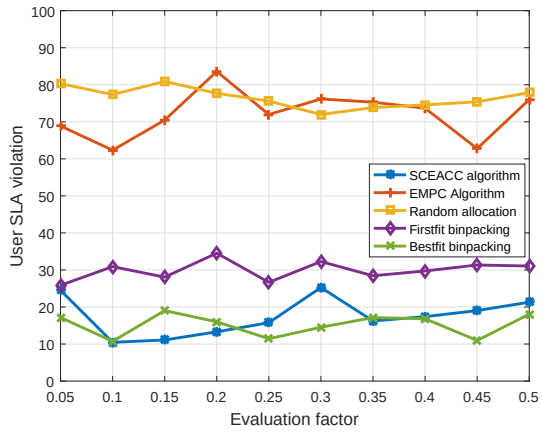


Figure 5.21: User's SLA violation rate

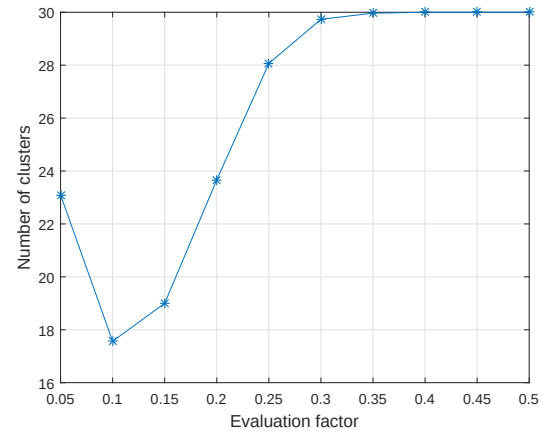


Figure 5.22: The number of clusters

**Scenario 3**: The evaluation factor $\delta$ is adjusted from 0.05 to 0.5, the squared coefficient of variation of service time $s_d^2$ is set to 1, and the maximum capacity of the data center $C(u)$ is adjusted 20000

# CHAPTER6. CONCLUSION AND FUTURE WORK

This thesis proposed cloud system operation improvement solutions by using mathematical optimization, queuing theory, algorithm design, and machine learning. In summary, we proposed efficient SDN and NFV deployment strategies in cloud computing infrastructure, that improve the operation of cloud computing infrastructure in terms of power consumption, processing time, communication time, and service quality.

In Chapter 3, we proposed an efficient big data distribution strategy in distributed heterogeneous cloud environments. The proposed solution minimizes data set processing time and cost simultaneously through a multi-objective linear programming model, and Pareto efficiency is obtained through the proposed algorithm. Based on simulation analysis, we were able to find multiple optimal solutions for the Pareto efficiency which are not dominated by other solutions.

In Chapter 4, an adaptive data center activation strategy is proposed based on user traffic prediction for power saving. The proposed strategy jointly controls the activation of network and server layer while assuring QoS and connectivity of the data center. The prediction model estimates a probability distribution of user demands by using Maximum Likelihood Estimation (MLE), and a Local Linear Regression (LLR) technique. The adaptive data center activation model is accomplished through Mixed Integer Linear Programming. In the experiment, the real traffic data collected from Google cluster is used for traffic prediction, and the data center activation is controlled depending on the prediction. Based on the observation of a one week period, the proposed algorithm saves an average of 47.7% of power compared to the full operation state of the data center.

In Chapter 5, NFV MANO strategy is proposed for each component. Virtual Infrastructure Manager (VIM) controls a single domain virtual infrastructure by scaling up and down VNFs. An NFV-SC assigned to a domain is modeled as a BCMP queuing network composed of open

networks and closed networks. Depending on user requests and interconnection between VNFs, the proposed algorithm maximizes the performance of service chains in the limited resource. NFV Orchestrator and VNF Manager controls the overall performance of NFV in multiple domains of infrastructure. For the efficient operation of NFV Orchestrator and VNF Manager, we proposed NFV-SC resource allocation and placement strategy in a hyper-scale cloud environment. NFV-SC is modeled as an M/G/1/K queuing network and the proposed strategy minimizes the power consumption of cloud infrastructure while guaranteeing a fair service quality for all service chains.

In future work, we would plan to extend the NFV deployment scheme to the online problem. We have studied the offline problem with given NFV-SC. However, NFV user requirements are highly variable, so fixed NFV-SC has limitations in providing scalable service to users. Extend the current research, we would like to study a scalable NFV-SC configuration and reconfiguration strategy depending on users requirements changes. We can increase or decrease the number of VNFs or reconfigure NFV-SC embedding for multiple objectives: power consumption, service time, or latency.

Also, the adaptive data center activation model can be extended to different data center architectures. We studied the power saving model only in Fat-tree data center architecture. However, there are a lot of data center architectures other than Fat-tree such as VL2, Bcube, and so on. We believe the adaptive activation model can be applied to other data center architectures by modifying the connectivity constraint in order to assure the seamless connectivity of the data center service.

# BIBLIOGRAPHY

[1] Y. Jadeja and K. Modi. *"Cloud computing - concepts, architecture and challenges."* Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on, Mar. 2012.

[2] C. C. Meixner, C. Develder, M. Tornatore. *"A Survey on Resiliency Technique in Cloud Computing Infrastructure and Applications."* IEEE Communications Surveys & Tutorials, vol. 18, Issue. 3, Feb. 2016.

[3] M. Guzek, P. Bouvry, E. G. Talbi. *"A Survey of Evolutionary Computation for Resource Management of Processing in Cloud Computing."* IEEE Computational Intelligence Magazine, vol. 10, Issue. 2, May. 2015.

[4] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J. M. Pierson, A. V. Vasilakos. *"Cloud Computing: Survey on Energy Efficiency."* Journal ACM Computing Surveys (CSUR), vol. 47, Issue. 2, Jan. 2015.

[5] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolk *"Software-Defined Networking: A Comprehensive Survey."* Proceedings of the IEEE, vol. 103, Issue. 1, Jan. 2015.

[6] R. Mijumbi, J.Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, R. Boutaba *"Network Function Virtualization: State-of-the-Art and Research Challenges."* IEEE Communications Surveys & Tutorials, vol. 18, Issue. 1, Sep. 2015.

[7] J. G. Herrera, J. F. Botero. *"Resource Allocation in NFV: A Comprehensive Survey."* IEEE Transactions on Network and Service Management. vol. 13, Issue. 3, Sep. 2016

[8] R. Jain, S. Paul. *"Network Virtualization and Software Defined Networking for Cloud Computing: a Survey."* IEEE Communications Magazine, vol. 51, Issue. 11, Nov. 2013

[9] M. Alicherry and T.V. Lakshman. *"Optimizing Data Access Latencies in Cloud Systems by Intelligent Virtual Machine Placement."* IEEE INFOCOM, 2013.

[10] M. Alicherry and T.V. Lakshman. *"Network Aware Resource Allocation in Distributed Clouds."* IEEE INFOCOM, 2012.

[11] X. Meng, V. Pappas, and Li Zhang. *"Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement."* IEEE INFOCOM, 2012.

[12] M. A. Sharkh, M. Jammal, A. Shami, and A. Ouda. *"Resource Allocation in a Network-Based Cloud Computing Environment: Design Challgnes."* IEEE Communization Magazine, November 2013.

[13] J. Doyle, R. Shorten, and D. OMahony. *"Stratus: Load Balancing the Cloud for Carbon Emissions Control."* IEEE Transactions on Cloud Computing, vol. 1, no. 1, pp.116128, 2013.

[14] T. Zidenberg, I. Keslassy, and U. Weiser. *"Optimal Resource Allocation with Multi-Amdahl."* IEEE Computer Architecture Letters, vol. 11, no. 2, pp.65-68, 2012.

[15] J. Cao, K. Li, and I. Stojmenovic. *"Optimal Power allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers."* IEEE Transaction on Computers, vol. 63, no. 1, 2014.

[16] A.Ruiz-Alvarez, M. Humphrey. *"A Model and Decision Procedure for Data Storage in Cloud Computing."* IEEE/ACM 12th International Symposium on Cluster, Cloud, and Grid Computing, 2012.

[17] Y. Li, M. Yao, and C. Lin. *"Joint Study on Optimizations of Data Center Deployment, VM Assignment and Migration."* IEEE/ACM 21st International Symposium on Quality of Service. 2013.

[18] R.T.Marler and J.S. Arora. *"Survey of Multi-Objective Optimizations Methods for Engineering."* Struct Multidisc Optim 26, pp.369-395,2014.

[19] Google Cloud price: https://developers.google.com/storage/pricing

[20] Amazon Cloud price: http://aws.amazon.com/ec2/pricing

[21] IBM Cloud: http://www.ibm.com/cloud-computing/us/en/iaas.html

[22] Rackspace Cloud price: http://www.rackspace.com/cloud/servers/pricing

[23] Microsoft Cloud price: http://azure.microsoft.com/en-us/pricing

[24] H. P. Benson. *"An Outer Approximation Algorithm for Generating All Efficient Extreme Points in the Outcome Set of Multi Objective Linear Programming Problem."* Journal of Global Optimization13 (1), pp 1-24, 1998.

[25] T. Zhang, K. Chen, C. Xu,G. Sun, T. Wang, and Y. Xie, *"Half-DRAM: a High-bandwidth and Lower-power DRAM Architecture from the Rethinking of Fine-grained Activation"*, Computer Architecture (ISCA) ACM/IEEE 41st International Sumposium on, pp.349-360, June 2014.

[26] W.-C. Chan and Y.-B. Lin *"Waiting time distribution for the M/M/m queue"*, Communications IEEE Proceedings, vol. 150, pp.159-162, June 2003.

[27] A. M. Hamad and A. E. Kamal*"Power Aware Connection Provisioning For All-Optical Multicast Traffic in WDM Networks"*, Optical Communication and Networking, IEEE/OSA Journals of, vol. 2,pp 481-495, July 2010.

[28] C. E. Leiserson*"Fat-trees: Universal networks for hardware-efficient supercomputing"*, Computers, IEEE Transactions on, vol. C-34, pp.892-901,Oct 1985.

[29] M. Al-Fares, A. Loukissas, A. Vahdat*"A Scalable, Commodity Data Center Network Architecture"*, Proceedings of the ACM SIGCOMM 2008, pp. 63-74, 2008.

[30] Y. Zhao, J. Zhang, Hui. Yang, and X. Yu*"Data center optimal networks (DCON) with OpenFlow based Software Defined Networking (SDN)"*, Communications and Networking in China (CHINACOM), pp. 771-775, Aug 2013.

[31] A. Sadasivarao, S. Syed, P. Pan, I. Monga, C. Guok, and A. Lake*"Bursting Data Between Data Centers: Case for Trnasport SDN"*, High-Performance Interconnects (HOTI), pp. 87-90, Aug 2013.

[32] D. Li, Y. Shang, and C. Chen*"Software Defined Green Data Center Network with Exclusive Routing"*, INFOCOM, 2014 Proceedings IEEE, pp. 1743-1751, April 2014.

[33] J. Cao, K. Li, and I. Stojmenovic*"Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers"*, Computers, IEEE Transactionson, vol. 63, pp. 45-58, May 2013.

[34] K. Zheng, X. Wang, L. Lie,and X. Wang*"Joint power optimization of data center network and servers with correlation analysis"*, INFOCOM, 2014 Proceedings IEEE, pp. 2598-2606, Apr. 2014.

[35] R. Ghosh, F. Longo, R. Xia, V. K. Naik, and K. S. Trivedi*"Stochastic Model Driven Capacity Planning for an Infrastructure-as-a-Service Cloud"*, Services Computing, IEEE Transactions on, vol. 7, pp. 667-680, Sep 2013.

[36] W. Ni, C. Huang, J. Wu*"Provisioning high-availability datacenter networks for full bandwidth communication"*, Communications and Networking in the Cloud, vol. 68, pp. 71-94, Aug 2014.

[37] Y. Zhang and N. Ansari*"HERO: Hierarchical Energy Optimization for Data Center Networks"*, Systems Journal, IEEE, pp. 1-10, Oct 2013.

[38] S. Paul, R. Jain, M. Samaka, and J. Pan*"Application delivery in multi-cloud environments using software defined networking"*, Communication and Networking in the Cloud, vol 68, pp. 166-186, Aug 2014.

[39] J. Whitney and P. Delforge, *"Data center efficiency assessment"*, Issue paper on NRDC (The Natural Resource Defense Council). August 2014.

[40] J. Fan, *"Local linear regression smoothers and their minmax efficiencies"*, Ann. Statist, vol 21, No. 1, pp 196-216. November 1993.

[41] E. Caron and F. Desprez, *"Forecasting for grid and cloud computing on demand resources based on pattern matching"*, Proceeding on IEEE Cloud Computing Technology and Science (CloudCom), pp 456-463. November 2010.

[42] M. Li and S. C. Lim, *"Modeling network traffic using generalized Cauchy process"*, Physica A: Statistical Mechanics and its application, vol 387, pp 2584-2594. April 2008.

[43] B. L. Dalmazo, J. P. Vilela, and M. Curado, *"Online traffic prediction in the cloud: a dynamic window approach"*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 9-14. August 2014.

[44] B. L. Dalmazo, J. P. Vilela, and M. Curado, *"Predicting traffic in the cloud: a statistical approach"*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 121-126. October 2013.

[45] A. A. Bankole and S. A. Ajila, *"Predicting cloud resource provisioning using machine learning techniques"*,Proceeding on IEEE Electrical and Computer Engineering (CCECE), pp 1-4. May 2013.

[46] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy, *"Profiling and modeling resource usage of virtualized application"*,Proceeding of the 9th ACM/IFIP/USENIX International conference on middleware, pp 386-387. December 2008.

[47] Google cluster-trace data, https://code.google.com/p/google clusterdata/

[48] M. Yoon, A. Kamal, Z. Zhu, *"Request prediction in cloud with a cyclic window learning algorithm"*, http://arxiv.org/abs/1507.02372.

[49] S. Andrade-Morelli, E. Ruiz-Snchez, E. Granell, J. Lloret, *"Energy consumption of wireless network access points, Green Communication and Networking"*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 113, 2013, pp 81-91.

[50] W. Fang, Z. Wang, J. LIoret, D. Zhang, Z. Yang, *"Optimising data placement and traffic routing for energy saving in Backbone Networks"*, Transactions on Emerging Telecommunications Technologies 25 (9), 914-925. 2014

[51] *"HP Altoline 6712 switch series data sheet"*, http://h20195.www2.hp.com/v2/default.aspx?cc=us &lc=en&oid=8566121.

[52] *"Cisco's Massively Scalable Data Center"*,http://www.cisco.com/ c/dam/en/us/td/docs/solutions/Enterprise/Data_Center/MSDC/ 10/MSDC_AAG_1.pdf.

[53] S. Chen, M. Ghorbani, Y. Wang, P. Bogdan, M. Pedram, *Trace-based analysis and prediction of cloud computing user behavior using the fractal modeling technique*, 2014 IEEE International Congress on Big Data, pp 733-739. June/July 2014.

[54] M. Lin, A. Weirman, L.L.H.Andres, E. Thereska, *"Dynamic right-sizing for power-proportional data centers"*, Proceedings on IEEE INFOCOM 2011, pp 1098-1106. April 2011.

[55] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, N. Pissinou , *"Joint Host-Network Optimization for Energy-Efficient Data Center Networking"*, in IPDPS, 2013.

[56] S. Islam, J. Keung, K. Lee, and A. Liu, *Empirical prediction models for adaptive resource provisioning in the cloud*, Future Generation Computer Systems, vol 28, pp 155-162,January 2012.

[57] B. L. Dalmazo, J. P. Vilela, and M. Curado, *Onlinetraffic prediction in the cloud: a dynamic window approach*, Proceeding on IEEE Cloud and Green Computing (CGC), pp 9-14, August 2014.

[58] J. Vilaplana, F. Solsona, I. Teixido, J. Mateo, F. Abella, J. Rius, *A queuing theory model for cloud computing*, The Journal of Supercomputing, vol. 69, no. 1, pp 492-507, April 2014.

[59] W.-H. Bai, J.-Q. Xi, J.-X. Zhu, S.-W. Huang, *Performance Analysis of Heterogeneous Data Centers in Cloud Computing Using a Complex Queuing Model*, Mathematical Problems in Engineering, vol. June 2015, Article ID 980945.

[60] J. Cao, K. Hwang, K. Li, A. Y. Zomaya, *Optimal multiserver configuration for profit maximization in cloud computing*, IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1087-1096, June 2013.

[61] *Green Abstraction Layer standard to manage energy consumption of telecom networks*, ETSI Standard (ES) 202 237, http://www.etsi.org/news-events/news/822-2014-09-news-green-abstraction-layer-standard-to-manage-energy-consumption-of-telecom-networks.

[62] Bolla, R. , ,Bruschi, R. , Davoli, F.a , Donadio, P. , Fialho, L. , Collier, M. , Lombardo, A. , Reforgiato, D. , Riccobene, V. , Szemethy, T., *A northbound interface for power management in next generation network devices*, IEEE Communications Magazine, vol. 52, Issue: 1, January 2014.

[63] Bolla, R. , ,Bruschi, R. , Davoli, F.a , Donadio, P. , Fialho, L. , Collier, M. , Lombardo, A. , Reforgiato, D. , Riccobene, V. , Szemethy, T., *The green abstraction layer: A standard power-management interface for next-generation network devices*, IEEE Internet Computing, vol. 17, No. 2, March - April 2013.

[64] S.C. Bruell, G. Balbo, and P.V. Afshari, *"Mean value analysis of mixed, multiple class BCMP networks with load dependent service stations"*, Performance Evaluation, 4, (1984), pp.241-260.

[65] K.M. Chandy and D. Neuse, *"Linearizer: A heuristic algorithm for queuing network models of computing systems"*, Magazine Communications of the ACM, vol 25, issue 2, Feb 1982. pp.126-134.

[66] A. Gemer-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, A. Akella, *"OpenNF: enabling innovation in network function control"*, Proceedings of the 2014 ACM conference on SIGCOMM, pp. 163-174.

[67] A. Basta, W. Kellerer, M. Hoffmann, H.J. Morper, and K. Hoffmann, *"Applying NFV and SDN to LTE mobile core gateways, the functions placement problem"*, in Proc. 4th Workshop AllThingsCellular, New York, NY, USA: ACM, 2014, pp. 33-38.

[68] M. Bouet, J. Leguay, and V. Conan, *"Cost-based placement of VDPI functions in nfv infrastructure"*, in Proc. IEEE 1st IEEE Conf. NetSoft, Apr. 2015, pp.1-9.

[69] S. Mehraghdam, M. Keller, and H. Karl, *"Specifying and placing chains of virtual network functions"*, in Proc. IEEE 3rd Int. Conf. CloudNet, Oct. 2014, pp. 7-13.

[70] M. xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, *"Network function placement for NFV chaining in packet/optical datacenters"*, J. Lightw. Technol., vol. 33, no. 8, pp. 1565-1570, Apr 2015.

[71] A. Zhang, P. Santos, D. Beyer, and H. Tang, *"Optimal Server Resource Allocation Using an Open Queuing Network Model of Response Time"*, available: http://www.hpl.hp.com/techreports/2002/HPL-2002-301.pdf.

[72] O. J. Boxma, *"Static Optimization of Queuing Systems"*, WSSIAA 5, 1995, pp. 1-16.

[73] R. Mijumbi, J. Serrat, J. gorricho, N. Bouten, and F. D. Turck, *"Network function virtualization: state-of the art and research challenges"*, IEEE Communications survey & tutorials, vol. 18, No. 1, first quarter 2016, pp. 236- 262.

[74] M. Reiser and S. S. Lavenberg, *"Mean-value analysis of closed multichain queuing networks"*, Journal of the association for computing machinery, vol. 27, No. 2, Apr 1980, pp. 313-322.

[75] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, *"Open, closed, and mixed networks of queues with different classes of customers"*, Journal of the association for computing machinery, vol. 22, No. 2, Apr. 1975, pp. 248-260.

[76] R. Szabo, M. Kind, F. J. Westphal, H. Woesner, d. Jocha, and A. Csaszar, *"Elastic network functions: opportunities and challenges"*, IEEE Network Magazine, vol. 29, No. 3, May/June 2015, pp. 15-29.

[77] W. ding, W. Qi, J. Wang, and B. Chen, *"OpenSCaaS: an open service chain as service platform toward the integration of SDN and NFV"*, IEEE Network Magazine, vol. 29, No. 3, May/June 2015, pp. 30-35.

[78] T. Wood, K.K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, *"Toward a software-based network: integrating software defined networking and network function virtualization"*, IEEE Network Magazine, vol. 29, No. 3, May/June 2015, pp. 36-41.

[79] R. Yu, G. Xue, V. T. Kilari, and X. Zhang, *"Network function virtualization in the multi-tenant cloud"*, IEEE Network Magazine, vol. 29, No. 3, May/June 2015, pp. 42-47.

[80] P.J. Schweitzer, G. Serazzi, M. Broglia, *"A queue-shift approximation technique for product-form queuing networks"*, Computer performance evaluation, vol. 1469 of the series lecture notes in computer science, pp. 267-279.

[81] Ander S. G. Andrae, T. Edler, *"On Global Electricity Usage of Communication Technology: Trends to 2030"*, Open Access Challanges 2015, 6(1), 117-157.

[82] European Telecommunication Standards Institute, *"Network Function Virtualisation-Update White Paper"*, SDN and OpenFlow World congress, Oct 15-17, 2013, https://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

[83] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M.Zhu, J. Zolla, U. Holzle, S. Stuart, and A. Vahdat, *"B4: experience with a globally-deployed software defined WAN"*, Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp.3-14, August 12-16 2013.

[84] J.M. Smith, *"Optimal design and performance modeling of M/G/1/K queuing systems"*, Journal Mathematical and Computer Modeling, Vol.30, Issue 9-10, pp.1049-1081, May 2004.

[85] J.M. Smith, *"Properties and performance modeling of finite buffer M/G/1/K networks"*, Journal Computers and Operations Research, Vol.38, Issue 4, pp.740-754, April 2011.

[86] F. Baccelli, B. Blaszczyszyn, *"Stochastic geometry and wireless networks, volume 1-theory"*, B. NoW Publishers, 1, pp.150, 2009, Foundations and Trends in Networking Vol. 3: No 3-4, pp 249-449 .

[87] Allen AO, *"Probability, statistics, and queuing theory with computer science applications"*, 2nd ed. Boston:Academic Press,Inc.;1990.

[88] M. Dayarathna, Y. Wen, R. Fan, *"Data center energy consumption modeling: a survey"*, IEEE Communications Surveys & Tutorials, Vol.18, Issue: 1, pp. 732-794, September 2015.

[89] H. Liu, C.Z. Xu, H. Jin, J. Gong, and X. Liao, *"Performance and energy modeling for live migration of virtual machines"*, in Proceedings of the 20th International Symposium on High Performance Distributed Computing, ser. HPDC'11. New York, NY, USA: ACM 2011, pp. 171-182.

[90] Z. Liu, Y. Chen, C. Bash, A. Wierman, G. Gmach, Z. Wang, M. Marwah, C. Hyser, *"Renewable and cooling aware workload management for sustainable data centers"*, in Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, ser. SIGMETRICS' 12. New York, NY, USA: ACM, 2012, pp.175-186.

[91] M. S. Yoon, A. E. Kamal, *"Power Minimization in Fat-Tree SDN Datacenter Operation"*, in Proceedings of IEEE Global Communications Conference (GLOBECOM) 2015, pp. 1-7.

[92] C. Shuchi, M. Konstantin, S. Tselil, Y. Grigory, *"Near Optimal LP Rounding Algorithm for CorrelationClustering on Complete and Complete k-partite Graphs"*, in Proceedings of 46th Annual ACM on Symposium on Theory of Computing.

[93] R. Mijumbi, J. Serrat, J.L. Gorricho, N. Bouten, F.D. Truct, R.Boutaba, *"Network Function Virtualization: State-of-the-Art and Research Challenges"*, IEEE Communications Surveys & Tutorials, vol. 18, Issue. 1, pp. 236-262, September 2015.

[94] E. Amaldi, S. Coniglio, A. M. C. A. Koster, T. Tieves, *"On the Computation Complexity of the Virtual Network Embedding Problems"*, Electron. Notes Discr. Math, vol 52, pp. 213-220, June 2016.

[95] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, H. Meer, *"Energy Efficient Virtual Network Embedding"*, IEEE Communications Letters, vol. 16, pp. 756-759, May 2012.

[96] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, H. Meer, *"Greener Networking in a Network Virtualization Environment"*, Journal Computer Networks: The International Journal of Computer Networks and Telecommunications Networking, vol. 57, pp. 2021-2039, June 2013.

[97] V. Eramo, E. Miucci, M. Ammar, *"Study of Reconfiguration Cost and Energy Aware VNE Policies in Cycle-Stationary Traffic Scenarios"*, IEEE Journal on Selected Areas in Communications, vol. 34, Issue. 5, pp. 1281-1297, May 2016.

[98] N.M.M.K. Chowdhury, M. R. Rahman, R. Boutaba, *"Virtual Network Embedding with Coordinated Node and Link Mapping"*, in Proc. IEEE Glob. Comm. Conf. Sel. Areas Commun. Softw. Defined Netw. Netw. Functions (GC-SAC-SDN), San Diego, CA, Usa, Dec. 2015, pp. 834-840.

[99] S. Liu, Z. Cai, H. Xu, M. Xu, *"Towards Security-Aware Virtual Network Embedding"*, Comput. Netw., vol. 91, pp. 151-163, Nov 2015.

[100] S. Liu, Z. Cai, H. Xu, M. Xu, *"Security-Aware Virtual Network Embedding"*, in Proc. IEEE Int. Conf. Commun. (ICC), Sydney, NSWm Austrailia, Jun 2014, pp. 834-840.

[101] M. R. Rahman, I. Aib, R. Boutaba, *"Survivable virtual network embedding"*, in: Networking, 2010, pp. 4052.

[102] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, L. Mathy, *Coordinated allocation of service function chains"*,in: Proceedings of the second ACM SIGCOMM Workshop on

Virtualized Infrastructure Systems and Architectures, VISA 10, ACM, New York, NY, USA, 2010, pp. 41 48.

[103] M. T. Beck, J. F. Botero, *"Adaptive virtual network provisioning"*,in Proc. IEEE Glob. Comm. Conf. Sel. Areas Commun. Softw. Defined Netw. Netw. Functions (GC-SAC-SDN), San Diego, CA, Usa, Dec. 2015, pp. 834-840.

[104] J. Elias, F. Martignon, S. Paris, J. Wang, *"Efficient Orchestration Mechanisms for Congestion Mitigation in NFV: Models and Algorithms"*,IEEE Trans. Service Comput., vol. 99, pp. 1-13, 2015.

[105] R. Brischi, A. Carrega, F. Davoli, *"A Game for Energy-Aware Allocation of Virtualized Network Functions"*,Journal of Electrical and Computer Engineering, vol. 2016, Article ID 4067186, 10 pages.

[106] S. Gu, Z. Li, C. Wu, C. Huang, *"An efficient auction mechanism for service chains in the NFV market"*,IEEE International Conference on INFOCOM 2016, April 2016.

[107] L. Gu, S. Tao, D. Zheng, H. Jin, *"Communication Cost Efficient Virtualized Network Function Placement for Big Data Processing"*, IEEE INFOCOM Confrence on Computer Communications Workshops (INFOCOM WKSHPS), Apr 2016.

[108] A. Aral, T. Ovatman, *"Network-aware Embedding of Virtual Machine Clusters onto Federated Cloud Infrastructure"*, The Journals on Systems and Software, vol. 120, Oct 2016, pp. 89-104.

[109] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, N. T. Hieu, H. Tenhunen, *"Energy-aware VM Consolidateion in Cloud Data Centers Using Utilization Prediction Model"*, IEEE Transactions on Cloud Computing, vol. pp, Issue. 99.

[110] V. Sekar, N. Egi, S. Ratnasamy, M.K. Reiter, G. Shi, *"Design and Implementation of a Consolidated Middlebox Architecture"*, in Proc, NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, Apr 25-27, 2012.

[111] R. Cohen, L. L. Eytan, J. S. Naor, D. Raz, *"Near Optimal Placement of Virtual Network Functions "*, IEEE INFOCOM Conference, Apr 2015.

[112] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievsky, M. Zhu, R. Ramanathan, et al, *"Onix: a Distributed Control Platform for Large-Scale Production Networks"*, in Proc, OSDI'10 Proceedings of the 9th USENIX conference on Operating Systems Design and Implementation, oct. 04, 2010, pp. 351-364.

[113] P. Xiao, W. Qu, H. Qi, Z. Li, Y. Xu, *"The SDN Controller Placement Problem for WAN"*, IEEE International Conference on Communications in China (ICCC) 2014, Oct. 2014.

[114] T. Wood, K. K. Ramkrishnan, J. Hwang, G. Liu, W. Zhang, *"Toward a Software-Based Network: Integrating Software Defined Networking and Network Function Virtualization"*, IEEE Network, vol. 29, Issue.3, June 2015, pp. 36-41.

[115] L. Kerbache, J.M. Smith, *"The Generalized Expansion Method for Open Finite Queuing Networks"*, European Journal of Operational Research, Elsevier, 1987, 32(3), pp. 448-461.

[116] Asmussen. S. R., *"Random Walks"*, Applied Probability and Queues. Stochastic Modeling and Applied Probability. 51. pp. 220-243.

[117] V. Persico, A. Botta, P. Marchetta, A. Montieri, A. Pescape, *"On the performance of the wide-are networks interconnecting public-cloud datacenters around the globe"*, Computer networks, Elsevier, vol. 112, Jan 2017, pp 67-83.